

311-CD-507-001

EOSDIS Core System Project

**Release 5A
Systems Management Subsystem
Database Design and
Schema Specifications
for the ECS Project**

**This document has not yet been approved by the
Government for general use or distribution.**

May 1999

Raytheon Systems Company
Upper Marlboro, Maryland

Release 5A Systems Management Subsystem Database Design and Schema Specifications for the ECS Project

May 1999

Prepared Under Contract NAS5-60000
CDRL Item #050

RESPONSIBLE ENGINEER

Maureen Muganda /s/ 5/27/99
Maureen Muganda Date
EOSDIS Core System Project

SUBMITTED BY

Mary S. Armstrong /s/ 5/27/99
Mary Armstrong, Development Engineering Manager Date
EOSDIS Core System Project

Raytheon Systems Company
Upper Marlboro, Maryland

This page intentionally left blank.

Preface

This document describes the data design and database specification for the Systems Management Subsystem. It is one of eight documents comprising the detailed database design specifications for each of the ECS subsystems.

The subsystem database design specifications for the as delivered system include:

311-CD-500 Release 5A Data Management (DM) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-501 Release 5A Ingest Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-502 Release 5A Interoperability Subsystem (IOS) Database Design and Database Schema Specifications for the ECS Project

311-CD-503 Release 5A Planning and Data Processing Subsystem (PDPS) Database Design and Database Schema Specifications for the ECS Project

311-CD-504 Release 5A Science Data Server (SDSRV) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-505 Release 5A Storage Management (STMGMT) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-506 Release 5A Subscription Server (SUBSRV) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-507 Release 5A Systems Management Subsystem (MSS) Database Design and Database Schema Specifications for the ECS Project

This submittal meets the milestone specified in the Contract Data Requirements List (CDRL) of NASA Contract NAS5-60000. It is a formal contract deliverable with an approval code 2. As such, it does not require formal Government approval, however, the Government reserves the right to request changes within 45 days of the initial submittal. Once approved, contractor changes to this document are handled in accordance with Class I and Class II change control requirements described in the EOS Configuration Management Plan, and changes to this document shall be made by Document Change Notice (DCN) or by complete revision.

Entity Relationship Diagrams (ERDs) presented in this document have been exported directly from tools and some cases contain too much detail to be easily readable within hard copy page constraints. The reader is encouraged to view these drawings on-line using the Portable Document Format (PDF) electronic copy available via the ECS Data Handling System (ECS) on the world-wide web at <http://edhs1.gsfc.nasa.gov>.

Any questions should be addressed to:

Data Management Office
The ECS Project Office
Raytheon Systems Company
1616 McCormick Drive
Upper Marlboro, Maryland 20774-5301

Abstract

This document outlines “as-built” database design and database schema of the Systems Management Subsystem database including the physical layout of the database and initial installation parameters.

Keywords: data, database, design, configuration, database installation, scripts, security, data model, data dictionary, replication, performance tuning, SQL server, database security, replication, database scripts

This page intentionally left blank.

Change Information Page

List of Effective Pages			
Page Number	Issue		
Title	Submitted as Final		
iii through xii	Submitted as Final		
1-1 and 1-2	Submitted as Final		
2-1 and 2-2	Submitted as Final		
3-1 through 3-68	Submitted as Final		
4-1 through 4-4	Submitted as Final		
5-1 through 5-4	Submitted as Final		
6-1 and 6-2	Submitted as Final		
7-1 through 7-6	Submitted as Final		
A-1 through A-6	Submitted as Final		
AB-1 and AB-2	Submitted as Final		
Document History			
Document Number	Status/Issue	Publication Date	CCR Number
311-CD-507-001	Final	May 1999	99-0466

This page intentionally left blank.

Contents

Preface

Abstract

1. Introduction

1.1	Identification	1-1
1.2	Scope	1-1
1.3	Purpose	1-1
1.4	Audience.....	1-1

2. Related Documents

2.1	Applicable Documents	2-1
2.2	Information Documents.....	2-2

3. Data Design

3.1	Database Overview.....	3-1
3.1.1	Physical Data Model Entity Relationship Diagram.....	3-1
3.1.2	Tables	3-2
3.1.3	Columns	3-11
3.1.4	Column Domains	3-29
3.1.5	Rules.....	3-29
3.1.6	Defaults	3-29
3.1.7	Views.....	3-29
3.1.8	Integrity Constraints	3-29
3.1.9	Triggers	3-29
3.1.10	Stored Procedures.....	3-51

3.2	Flat File Usage.....	3-55
3.2.1	File Descriptions	3-56
3.2.2	Block Specifications.....	3-57
3.2.3	Field Specifications	3-58
3.2.4	Domain Definitions	3-59

4. Performance and Tuning Factors

4.1	Indexes	4-1
4.2	Segments	4-3
4.3	Caches	4-3

5. Database Security

5.1	Approach	5-1
5.4	Login/Group Object Permissions	5-3

6. Scripts

6.1	Installation Scripts.....	6-1
6.2	De-Installation Scripts.....	6-1
6.3	Backup and Recovery Scripts.....	6-1
6.4	Miscellaneous Scripts.....	6-2

7. Replication

7.1	Replication Overview.....	7-1
7.2	Replication Definitions	7-2
7.3	Replication Subscriptions.....	7-4
7.4	Replication Database Configuration	7-5
7.5	Replication Server Configuration.....	7-5

List of Figures

3-1	Sample ERD	3-1
5-1	Sybase General Approach to SQL Server Security	5-1

List of Tables

3-1	Data Table Listing	3-2
3-2	EcAcOrder.....	3-3
3-3	EcAcOrderId	3-4
3-4	EcAcRequest	3-4
3-5	EcAcRequestId	3-5
3-6	EcDbVersions.....	3-5
3-7	L_LOCAL_DAAC	3-6
3-8	MsAcAffiliationCode.....	3-6
3-9	MsAcAsterCategory.....	3-6
3-10	MsAcDAACCode	3-6
3-11	MsAcInternetAffiliationCode	3-7
3-12	MsAcMediaFormatCode.....	3-7
3-13	MsAcMediaTypeCode	3-7
3-14	MsAcPriorityCode.....	3-7
3-15	MsAcResearchFieldCode.....	3-8
3-16	MsAcStatusCode.....	3-8
3-17	MsAcUsrProfile	3-8
3-18	MsAcUsrRequest	3-10
3-19	role_to_cots	3-11
3-20	Trigger Listing.....	3-30
3-21	Procedure Listing	3-51
3-22	Flat File Descriptions	3-56

3-23	Flat File Block Descriptions.....	3-57
3-24	Flat File Field Specifications.....	3-58
3-25	Flat File Domain Definitions.....	3-59
4-1	Index Type Key	4-1
4-2	Index List.....	4-2
4-3	Segment Descriptions.....	4-3
5-1	Permission Key.....	5-3
5-2	Group/Role Assignments	5-3
5-3	Object Permissions.....	5-4
6-1	Installation Scripts.....	6-1
6-2	De-Installation Scripts.....	6-1
6-3	Backup and Recovery Scripts.....	6-2
6-4	Miscellaneous Scripts and Input Data Files	6-2

Appendix A. MSS ERDs

Abbreviations and Acronyms

1. Introduction

1.1 Identification

This Systems Management Subsystem (MSS) Accountability Database Design and Database Schema Specification document, Contract Data Requirement List (CDRL) Item Number 050, whose requirements are specified in Data Item Description (DID) 311/DV2, is a required deliverable under the Earth Observing System (EOS) Data and Information System (EOSDIS) Core System (ECS), Contract NAS5-60000.

1.2 Scope

The MSS Accountability Database Design and Database Schema Specification document describes the data design and database specifications to support the data requirements of Release 5A MSS software.

1.3 Purpose

The purpose of the MSS Accountability Database Design and Database Schema Specification document is to support the maintenance of MSS data and databases throughout the life cycle of ECS. This document communicates the database implementation in sufficient detail to support ongoing configuration management.

1.4 Audience

This document is intended to be used by ECS maintenance and operations staff. The document is organized as follows:

Section 1 provides information regarding the identification, purpose, scope and audience of this document.

Section 2 provides a listing of the related documents, which were used as a source of information for this document.

Section 3 contains the MSS physical data model which is the database tables, triggers, stored procedures, and flat files.

Section 4 provides a description of database performance and tuning features such as indexes, caches, and data segments.

Section 5 provides a description of the security infrastructure used, and a list of the users, groups, and permissions available upon initial installation.

Section 6 provides a description of database and database related scripts used for installation, de-installation, backup/recovery, and other miscellaneous functions.

Section 7 contains replication design and implementation details.

2. Related Documents

2.1 Applicable Documents

The following documents, including Internet links, are referenced in this document, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this volume.

305-CD-500	Release 5A Segment Design Specification for the ECS Project
920-TDG-009	DAAC Hardware Database Mapping/GSFC
920-TDN-009	DAAC Hardware Database Mapping/NSIDC
920-TDE-009	DAAC Hardware Database Mapping/EDC
920-TDL-009	DAAC Hardware Database Mapping/LARC
920-TDS-009	DAAC Hardware Database Mapping/SMC
920-TDG-010	DAAC Database Configuration/GSFC
920-TDN-010	DAAC Database Configuration/NSIDC
920-TDE-010	DAAC Database Configuration/EDC
920-TDL-010	DAAC Database Configuration/LARC
920-TDS-010	DAAC Database Configuration/SMC
920-TDG-011	DAAC Sybase Log Mapping/GSFC
920-TDN-011	DAAC Sybase Log Mapping/NSIDC
920-TDE-011	DAAC Sybase Log Mapping/EDC
920-TDL-011	DAAC Sybase Log Mapping/LARC
920-TDS-011	DAAC Sybase Log Mapping/SMC
922-TDG-013	Disk Partitions/GSFC
922-TDN-013	Disk Partitions/NSIDC
922-TDE-013	Disk Partitions/EDC
922-TDL-013	Disk Partitions/LARC
922-TDS-013	Disk Partitions/SMC

These documents are maintained as part of the ECS baseline and available on the world-wide web at the URL: <http://cmdm.east.hitc.com/baseline>. Please note that this is a partial mirror site in that some items are not available (they are identified) since this is OPEN to all. This site may also be reached through the EDHS homepage. Scroll page to the connections line and click on the ECS Baseline Information System link.

2.2 Information Documents

The following documents, although not directly applicable, amplify or clarify the information presented in this document. These documents are not binding on this document.

313-CD-500	Release 5A CSMS/SDPS Internal ICD for the ECS Project
609-CD-500	Release 5A Operations Tools Manual for the ECS Project
611-CD-500	Release 5A Mission Operation Procedures for the ECS Project

These documents are accessible via the EDHS homepage.

3. Data Design

3.1 Database Overview

The MSS database implements the large majority of the persistent data requirements for the MSS Accountability Management Service CSC. The database is designed in such a manner as to satisfy business policy while maintaining data integrity and consistency. Database tables are implemented using the Sybase Relational Database Management system (DBMS). All components of the MSS database are described in the sections which follow, in sufficient detail to support maintenance needs.

3.1.1 Physical Data Model Entity Relationship Diagram

The Entity Relationship Diagram (ERD) presents a schematic depiction of the MSS physical data model. The ERDs presented here for the MSS database were produced using the S-Designer Data Architect Computer Aided Software Engineering (CASE) tool. ERDs represent the relationship between entities or database tables. On ERDs, tables are represented by rectangles and relationships are represented as arrows (see Figure 4-1).

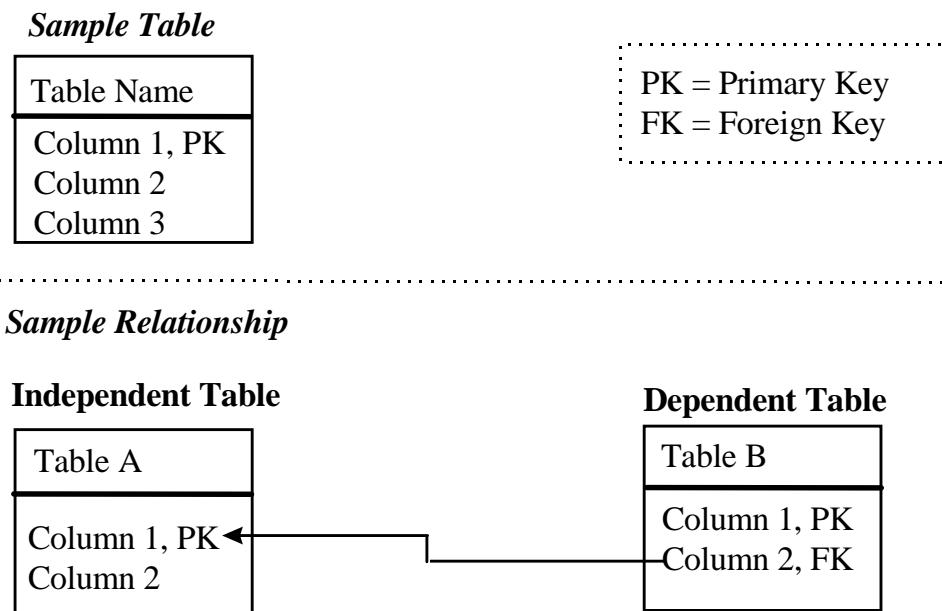


Figure 3-1. Sample ERD

The ERDs for the MSS database are found in the Appendix.

3.1.2 Tables

A listing of each of the tables in the MSS is given here. A brief definition of each of these tables follows including a listing of the columns comprising the table. The Column List indicates if the column is part of the primary key (PK) for the table. That is if the columns can be used alone or in combination with other primary key columns to uniquely identify a single row in the table. The column list also indicates whether the column is a mandatory column that must be included in every row.

Data requirements for MSS fall into five logical categories:

Order information - orders and requests placed for ECS products

Site information – information on the sites at which the database is housed

User Data – registered user information, user audit data, and user registration requests

Validation data – Domain definitions for codes used by MSS software

Versioning information – database schema version data

Table 3-1. Data Table Listing

Table Name	Contents	Logical Grouping
EcAcOrder	Order Information	Order Information
EcAcOrderId	Next available order ID	Order Information
EcAcRequest	Request Information	Order Information
EcAcRequestId	Next available request ID	Order Information
EcDbVersions	The current schema version of the MSS database.	Versioning Information
L_LOCAL_DAAC	Information on the local DAAC site housing this database.	Site Information
MsAcAffiliationCode	Affiliation Codes	Validation Data
MsAcAsterCategory	Valid Aster categories.	Validation Data
MsAcDAACCCode	Valid DAAC shortnames	Validation Data
MsAcInternetAffiliationCode	Internet affiliation codes	Validation Data
MsAcMediaFormatCode	Valid media formats for an order	Validation Data
MsAcMediaTypeCode	Valid media type codes	Validation Data
MsAcPriorityCode	Valid order/request priority codes	Validation Data
MsAcResearchFieldCode	Valid research field codes	Validation Data
MsAcStatusCode	Valid order/request status codes.	Validation Data
MsAcUsrAudit	Auditing information	User Data
MsAcUsrProfile	User information	User Data
MsAcUsrRequest	User request information	User Data
role_to_cots	Role an operational user provides for a COTS.	User Data

Table 3-2. EcAcOrder

Description

This table stores the end-user's order information. The data is used to fill and ship the request and to track the status of the order. An order can have many requests, and it may not be associated with a user in the MsAcUsrProfile table. (i.e. the order may 'belong' to a guest user). Data is stored indefinitely in the table.

Column List

Name	Code	Type	PK	Mandatory
abortedFlag	abortedFlag	char(1)	No	No
cancelledFlag	cancelledFlag	char(1)	No	No
eMailAddr	eMailAddr	varchar(255)	No	No
finishDateTime	finishDateTime	smalldatetime	No	No
firstName	firstName	varchar(20)	No	No
homeDAAC	homeDAAC	varchar(10)	No	No
lastName	lastName	varchar(20)	No	No
middleInit	middleInit	char(1)	No	No
orderDesc	orderDesc	varchar(50)	No	No
orderDistFormat	orderDistFormat	varchar(64)	No	No
orderGranule	orderGranule	numeric(9)	No	No
orderHomeDAAC	orderHomeDAAC	varchar(10)	No	Yes
orderId	orderId	varchar(10)	No	Yes
orderMedia	orderMedia	varchar(20)	No	No
orderPriority	orderPriority	varchar(10)	No	No
orderSize	orderSize	numeric(9)	No	No
orderSource	orderSource	varchar(12)	No	No
orderStatus	orderStatus	varchar(22)	No	No
receiveDateTime	receiveDateTime	smalldatetime	No	No
shipAddrCity	shipAddrCity	varchar(35)	No	No
shipAddrCountry	shipAddrCountry	varchar(30)	No	No
shipAddrFax	shipAddrFax	varchar(22)	No	No
shipAddrPhone	shipAddrPhone	varchar(22)	No	No
shipAddrState	shipAddrState	varchar(20)	No	No
shipAddrStreet1	shipAddrStreet1	varchar(35)	No	No
shipAddrStreet2	shipAddrStreet2	varchar(35)	No	No
shipAddrZip	shipAddrZip	varchar(15)	No	No
shipDateTime	shipDateTime	smalldatetime	No	No
startDateTime	startDateTime	smalldatetime	No	No
timeOfLastUpdate	timeOfLastUpdate	smalldatetime	No	No
title	title	varchar(5)	No	No
userId	userId	varchar(12)	No	No

Table 3-3. EcAcOrderId

Description

This table is used to generate the next orderId for the DAAC identified in the L_LOCAL_DAAC table. At any given time, there must be only one row in this table.

Column List

Name	Code	Type	PK	Mandatory
orderId	orderId	numeric(10)	No	No

Table 3-4. EcAcRequest (1 of 2)

Description

This table stores the shipping and tracking data for an end-user's request. A user can place one or many requests for a given order.

Column List

Name	Code	Type	PK	Mandatory
destinationDirectory	destinationDirectory	varchar(20)	No	No
destinationNode	destinationNode	varchar(20)	No	No
deviceDensity	deviceDensity	varchar(20)	No	No
deviceId	deviceId	varchar(20)	No	No
eMailAddr	eMailAddr	varchar(255)	No	No
ESDT_Id	ESDT_Id	varchar(20)	No	No
finishDateTime	finishDateTime	Smalldatetime	No	No
firstName	firstName	varchar(20)	No	No
ftpAddress	ftpAddress	varchar(128)	No	No
ftpPassword	ftpPassword	varchar(16)	No	No
lastName	lastName	varchar(20)	No	No
mediaType	mediaType	varchar(20)	No	No
middleInit	middleInit	char(1)	No	No
numBytes	numBytes	numeric(9)	No	No
numFiles	numFiles	numeric(9)	No	No
numGranule	numGranule	numeric(9)	No	No
orderHomeDAAC	orderHomeDAAC	varchar(10)	No	Yes
orderId	orderId	varchar(10)	No	Yes
parentId	parentId	varchar(10)	No	No
receiveDateTime	receiveDateTime	Smalldatetime	No	No
requestDesc	requestDesc	varchar(50)	No	No
requestDistFormat	requestDistFormat	varchar(64)	No	No
requestHomeDAAC	requestHomeDAAC	varchar(12)	No	Yes
requestId	requestId	varchar(10)	No	Yes

Table 3-4. EcAcRequest (2 of 2)

Name	Code	Type	PK	Mandatory
requestPriority	requestPriority	varchar(10)	No	No
requestStatus	requestStatus	varchar(22)	No	No
shipAddrCity	shipAddrCity	varchar(35)	No	No
shipAddrCountry	shipAddrCountry	varchar(30)	No	No
shipAddrFax	shipAddrFax	varchar(22)	No	No
shipAddrPhone	shipAddrPhone	varchar(22)	No	No
shipAddrState	shipAddrState	varchar(20)	No	No
shipAddrStreet1	shipAddrStreet1	varchar(35)	No	No
shipAddrStreet2	shipAddrStreet2	varchar(35)	No	No
shipAddrZip	shipAddrZip	varchar(15)	No	No
shipDateTime	shipDateTime	Smalldatetime	No	No
startDateTime	startDateTime	Smalldatetime	No	No
tapeFormat	tapeFormat	varchar(20)	No	No
timeOfLastUpdate	timeOfLastUpdate	Smalldatetime	No	No
title	title	varchar(5)	No	No

Table 3-5. EcAcRequestId**Description**

This table is used to generate the next requestId for the DAAC identified in the L_LOCAL_DAAC table. At any given time, there must be only one row in this table.

Column List

Name	Code	Type	PK	Mandatory
requestId	requestId	numeric(10)	No	No

Table 3-6. EcDbVersions

Description: This identifies the current version level of the MSS database

Column List

Name	Type	P	M
EcDbSchemaVersionID	Smallint	Yes	Yes
EcDbDropVersion	Char(64)	No	Yes
EcDbDropDescription	Varchar(255)	No	Yes
EcDbCurrentVersionFlag	Char(1)	No	Yes
EcDbDatabaseName	Varchar(255)	No	No
EcDbDropInstallDate	Datetime	No	No
EcDbSybaseVersion	Varchar(255)	No	No
EcDbSybaseServer	Varchar(255)	No	No
EcDbComments	Varchar(255)	No	No
EcDbUpdateProcess	Varchar(255)	No	No

Table 3-7. L_LOCAL_DAAC

Description

This table holds the database's DAAC short and long name. This table **must** have only one row, and **must** be the identifier of the DAAC at which the database is installed.

Column List

Name	Code	Type	PK	Mandatory
daac_name	daac_name	varchar(50)	No	Yes
daac_short	daac_short	char(10)	Yes	Yes

Table 3-8. MsAcAffiliationCode

Description

This is a lookup table, it defines the list of user affiliations.

Column List

Name	Code	Type	PK	Mandatory
AffiliationCode	AffiliationCode	varchar(16)	Yes	Yes
AffiliationDesc	AffiliationDesc	varchar(255)	No	No

Table 3-9. MsAcAsterCategory

Description:

This is a lookup table, it defines the list of Aster categories.

Column List

Name	Code	Type	PK	Mandatory
asterCategory	asterCategory	varchar(40)	No	No
asterCategoryId	asterCategoryId	numeric(2)	Yes	Yes

Table 3-10. MsAcDAACCCode

Description

This is a lookup table, it lists all the DAACs abbreviations and names.

Column List

Name	Code	Type	PK	Mandatory
DAACAbbrv	DAACAbbrv	varchar(3)	Yes	Yes
DAACLongName	DAACLongName	varchar(255)	No	No
DAACShortName	DAACShortName	varchar(10)	No	Yes

Table 3-11. MsAcInternetAffiliationCode

Description

This is a lookup table, it list all the internet affiliations.

Column List

Name	Code	Type	PK	Mandatory
InternetAffiliationCode	InternetAffiliationCode	varchar(14)	Yes	Yes
InternetAffiliationDesc	InternetAffiliationDesc	varchar(255)	No	No

Table 3-12. MsAcMediaFormatCode

Description

This is a lookup table, it list all the available media format.

Column List

Name	Code	Type	PK	Mandatory
MediaFormatCode	MediaFormatCode	varchar(20)	Yes	Yes
MediaFormatDesc	MediaFormatDesc	varchar(255)	No	No

Table 3-13. MsAcMediaTypeCode

Description

This is a lookup table, it list all the media type available.

Column List

Name	Code	Type	PK	Mandatory
MediaTypeCode	MediaTypeCode	varchar(20)	Yes	Yes
MediaTypeDesc	MediaTypeDesc	varchar(255)	No	No

Table 3-14. MsAcPriorityCode

Description

This is a lookup table, it defines the list of user request priority levels.

Column List

Name	Code	Type	PK	Mandatory
PriorityCode	PriorityCode	varchar(10)	Yes	Yes
PriorityDesc	PriorityDesc	varchar(255)	No	No

Table 3-15. MsAcResearchFieldCode

Description

This is a lookup table, it defines the list of user research fields.

Column List

Name	Code	Type	PK	Mandatory
ResearchFieldCode	ResearchFieldCode	varchar(64)	Yes	Yes
ResearchFieldDesc	ResearchFieldDesc	varchar(255)	No	No

Table 3-16. MsAcStatusCode

Description

This is a lookup table, it defines the list order statuses.

Column List

Name	Code	Type	PK	Mandatory
StatusCode	StatusCode	varchar(22)	Yes	Yes
StatusDesc	StatusDesc	varchar(255)	No	No

Table 3-17. MsAcUsrProfile (1 of 2)

Description

This table stores identifying, authenticating, and other data that is used by ECS servers to distribute data to registered users.

Column List

Name	Code	Type	PK	Mandatory
accountNumber	accountNumber	varchar(17)	No	No
affiliation	affiliation	varchar(16)	No	No
asterCategory	asterCategory	numeric(2)	No	No
billAddrCity	billAddrCity	varchar(35)	No	No
billAddrCountry	billAddrCountry	varchar(30)	No	No
billAddrFax	billAddrFax	varchar(22)	No	No
billAddrPhone	billAddrPhone	varchar(22)	No	No
billAddrState	billAddrState	varchar(20)	No	No
billAddrStreet1	billAddrStreet1	varchar(35)	No	No
billAddrStreet2	billAddrStreet2	varchar(35)	No	No
billAddrZip	billAddrZip	varchar(15)	No	No
creationDate	creationDate	smalldatetime	No	No

Table 3-17. MsAcUsrProfile (2 of 2)

Name	Code	Type	PK	Mandatory
darExpeditedData	darExpeditedData	bit	No	Yes
ECSAuthenticator	ECSAuthenticator	varchar(32)	No	Yes
eMailAddr	eMailAddr	varchar(255)	No	No
expirationDate	expirationDate	smalldatetime	No	No
firstName	firstName	varchar(20)	No	Yes
GTWYUsrType	GTWYUsrType	varchar(20)	No	No
homeDAAC	homeDAAC	varchar(10)	No	Yes
internetAffiliation	internetAffiliation	varchar(14)	No	No
lastName	lastName	varchar(20)	No	Yes
mailAddrCity	mailAddrCity	varchar(35)	No	No
mailAddrCountry	mailAddrCountry	varchar(30)	No	No
mailAddrFax	mailAddrFax	varchar(22)	No	No
mailAddrPhone	mailAddrPhone	varchar(22)	No	No
mailAddrState	mailAddrState	varchar(20)	No	No
mailAddrStreet1	mailAddrStreet1	varchar(35)	No	No
mailAddrStreet2	mailAddrStreet2	varchar(35)	No	No
mailAddrZip	mailAddrZip	varchar(15)	No	No
middleInit	middleInit	char(1)	No	No
motherMaidenName	motherMaidenName	varchar(20)	No	No
nasaUser	nasaUser	char(1)	No	No
organization	organization	varchar(31)	No	No
privilegeLevel	privilegeLevel	varchar(10)	No	No
projectName	projectName	varchar(30)	No	No
researchField	researchField	varchar(64)	No	No
shipAddrCity	shipAddrCity	varchar(35)	No	No
shipAddrCountry	shipAddrCountry	varchar(30)	No	No
shipAddrFax	shipAddrFax	varchar(22)	No	No
shipAddrPhone	shipAddrPhone	varchar(22)	No	No
shipAddrState	shipAddrState	varchar(20)	No	No
shipAddrStreet1	shipAddrStreet1	varchar(35)	No	No
shipAddrStreet2	shipAddrStreet2	varchar(35)	No	No
shipAddrZip	shipAddrZip	varchar(15)	No	No
telNum	telNum	varchar(22)	No	No
title	title	varchar(5)	No	No
userId	userId	varchar(12)	No	Yes

Table 3-18. MsAcUsrRequest (1 of 2)**Description**

This table stores request for user registration. When a user places a request, the user must be approved first. If a user is approved, the user's data is copied into the MsAcUsrProfile, and deleted from this table, if it is denied, the user's data remains in this table until it is deleted by the operator.

Column List

Name	Code	Type	PK	Mandatory
accountNumber	accountNumber	varchar(17)	No	No
affiliation	affiliation	varchar(16)	No	No
billAddrCity	billAddrCity	varchar(35)	No	No
billAddrCountry	billAddrCountry	varchar(30)	No	No
billAddrFax	billAddrFax	varchar(22)	No	No
billAddrPhone	billAddrPhone	varchar(22)	No	No
billAddrState	billAddrState	varchar(20)	No	No
billAddrStreet1	billAddrStreet1	varchar(35)	No	No
billAddrStreet2	billAddrStreet2	varchar(35)	No	No
billAddrZip	billAddrZip	varchar(15)	No	No
creationDate	creationDate	smalldatetime	No	No
eMailAddr	eMailAddr	varchar(255)	No	No
expirationDate	expirationDate	smalldatetime	No	No
firstName	firstName	varchar(20)	No	Yes
homeDAAC	homeDAAC	varchar(10)	Yes	Yes
lastName	lastName	varchar(20)	No	Yes
mailAddrCity	mailAddrCity	varchar(35)	No	No
mailAddrCountry	mailAddrCountry	varchar(30)	No	No
mailAddrFax	mailAddrFax	varchar(22)	No	No
mailAddrPhone	mailAddrPhone	varchar(22)	No	No
mailAddrState	mailAddrState	varchar(20)	No	No
mailAddrStreet1	mailAddrStreet1	varchar(35)	No	No
mailAddrStreet2	mailAddrStreet2	varchar(35)	No	No
mailAddrZip	mailAddrZip	varchar(15)	No	No
middleInit	middleInit	char(1)	No	No
motherMaidenName	motherMaidenName	varchar(20)	No	No
nasaUser	nasaUser	char(1)	No	No
organization	organization	varchar(31)	No	No
privilegeLevel	privilegeLevel	varchar(10)	No	No
projectName	projectName	varchar(30)	No	No
researchField	researchField	varchar(64)	No	No
shipAddrCity	shipAddrCity	varchar(35)	No	No
shipAddrCountry	shipAddrCountry	varchar(30)	No	No
shipAddrFax	shipAddrFax	varchar(22)	No	No
shipAddrPhone	shipAddrPhone	varchar(22)	No	No
shipAddrState	shipAddrState	varchar(20)	No	No
shipAddrStreet1	shipAddrStreet1	varchar(35)	No	No
shipAddrStreet2	shipAddrStreet2	varchar(35)	No	No
shipAddrZip	shipAddrZip	varchar(15)	No	No

Table 3-18. MsAcUsrRequest (2 of 2)

Name	Code	Type	PK	Mandatory

Name	Code	Type	PK	Mandatory
status	status	varchar(7)	No	No
telNum	telNum	varchar(22)	No	No
title	title	varchar(5)	No	No
userId	userId	varchar(12)	Yes	Yes

Table 3-19. role_to_cots

Description

This defines operator's roles with accessible cots.

Column List

Name	Code	Type	PK	Mandatory
cots_list	cots_list	varchar(36)	No	No
roleID	roleID	varchar(15)	Yes	Yes

3.1.3 Columns

Brief definitions of each of the columns present in the database tables defined above are contained herein.

Column: abortedFlag

Description

This column indicates whether an order has been aborted.

Valid Values:

Y=Yes
N=No

Column: accountNumber

Description

This is the account number that is given by the user when they request a user profile. It becomes associated with the user profile.

Column: affiliation

Description

This column contains the user's affiliation.

Valid Values:

- Gov. Research
- Government Other
- Univ. Research
- Univ. Class Work
- Commercial
- Kinder.-12 Grade

Column: AffiliationCode

Description

This is the user's affiliation code.

Valid Values:

- Gov. Research
- Government Other
- Univ. Research
- Univ. Class Work
- Commercial
- Kinder.-12 Grade

Column: AffiliationDesc

Description

This column contains the long description of the affiliation code.

Column: asterCategory

Description

MsAcUsrProfile: This column contains the user's aster category id.

MsAcAsterCategory: The is the description of the aster category id.

Column: asterCategoryId

Description

This column defines an aster category identifier.

Valid Values:

- | | |
|---|-----------------------|
| 0 | Not an ASTER DAR User |
| 1 | MITI/NASA |

2	EOS Member
3	IEOS agencies
4	ASTER Science Team Leader
5	US Team Leader
6	ASTER Science Working Group
7	ASTER Science Team Member
8	AO User
9	Special-Priority Japan User
10	EOS Science Project Office
11	ASTER Science Project
12	ASTER CDS/ESDIS Project
13	ASTER Instrument Team
14-99	Category 14-99

Column: billAddrCity

Description

This is the user's city, for billing purposes.

Column: billAddrCountry

Description

This is the user's country, for billing purposes.

Column: billAddrFax

Description

This is the user's fax number, for billing purposes.

Column: billAddrPhone

Description

This is the user's phone number, for billing purposes.

Column: billAddrState

Description

This is the user's state address, for billing purposes.

Column: billAddrStreet1

Description

This is the user's street address, for billing purposes.

Column: billAddrStreet2

Description

This is the user's street address, for billing purposes. Used only if address street is longer than what can be accommodated in billAddrStreet1.

Column: billAddrZip

Description

This is the user's zip code address, for billing purposes.

Column: cancelledFlag

Description

This column identifies whether an order has been cancelled.

Valid Values:

Y=Yes
N=No

Column: cots_list

Description

A list of cots accessible by a roleID.

Column: creationDate

Description

This is the date that the userid was created.

Column: currentVersion

Description

This is the current version of the database schema.

Column: daac_name

Description

This is the DAAC's long name.

Valid Values:

Alaska SAR Facility

Consortium for International Earth Science Information Network
EROS Data Center
Goddard Space Flight Center
Jet Propulsion Laboratory
Langley Research Center
National Snow and Ice Data Center
Oak Ridge National Laboratory

Column: daac_short

Description

This is the short home DAAC name.

Valid Values:

ASF
CSN
EDC
GSF
JPL
LAR
NSC
ORN

Column: DAACAbbrv

Description

This is the 3-letters name abbreviation of the DAAC's (same values as daac_short).

Column: DAACLongName

Description

This is the DAAC's long name. (same values as daac_name)

Column: DAACShortName

Description

This is the short name abbreviation of the DAAC's (same values as daac_short).

Column: darExpeditedData

Description

This column is false if the user is not allowed to submit DARs that request expedited data. The column is true if the user is allowed to submit DARs that request expedited data

Valid Values:

0=False
1=True

Column: destinationDirectory

Description

This column holds the user's destination directory for ftp acquires.

Column: destinationNode

Description

This column holds the user's destination node for ftp acquires.

Column: deviceDensity

Description

This column holds the request's device density.

Column: deviceId

Description

This column holds the requests's device ID.

Column: EcDbComment

Description:

Notes or comments on the database version level.

Column: EcDbCurrentVersionFlag

Description:

Flag indicating if this row represents the current database version entry

Valid Values: 1= yes, 0 = no

Column: EcDbDatabaseName

Description:

The name of the database for which this database version level is applied.

Column: EcDbDropDescription

Description:

The official description of the ECS software drop for this database version level.

Column: EcDbDropInstallDate

Description:

The date and time that the database version level was installed.

Column: EcDbDropVersion

Description:

The official name of the ECS software drop for this database version level.

Column: EcDbSchemaVersionId

Description:

The subsystem-specific identifier for this database schema version.

Column: EcDbSybaseServer

Description:

The name of the baseline Sybase SQL server controlling this database.

Valid Values: See 920-TDx-009

Column: EcDbSybaseVersion

Description:

The software release version of the Sybase SQL server in place when this database version level was initially installed.

Column: EcDbUpdateProcess

Description:

The installation method by which this database version level was installed

Column: ECSAuthenticator

Description

Authentication entry for the user used to authenticate access.

Column: eMailAddr

Description

This is the user's email address.

Column: ESDT_Id

Description

This name will identify the short name associated with the collection or granule.

Valid Values:

Refer to Document 910-TDA-019 ESDT Baseline

Column: expirationDate

Description

This is the expiration date of the user's profile. (i.e. account)

Column: files

Description

This is the number of files.

Column: filesystem

Description

This is the file system identifier.

Column: finishDateTime

Description

EcAcOrder: The column contains the time when all requests for the order have been completed.

EcAcRequest: This column holds the time set by DDIST to the time DDIST finished writing the media for the request (i.e., the request status changed to "waiting for shipment").

Column: firstName

Description

This is the user's first name.

Column: ftpAddress

Description

This column holds a request's ftp staging address.

Column: ftpPassword

Description

This column holds the ftp password for the staging request.

Column: GTWYUsrType

Description

For registered users, the gateway will retrieve their user profile and check this attribute. If it is filled, it will use GTWYUsrType and a generated password to log the user into DCE (rather than the userID attribute). A DCE account for GTWYUsrType must exist with the current V0GwPwd as its password.

Valid Values:

DAACOPS	DAAC Operations User
ECSDEV	ECS Development User
V0CERES	V0 CERES User
GUEST	Guest User

Column: homeDAAC

Description

This is the name of a DAAC, this is where the Request was issued.

Column: internetaffiliation

Description

The column contains the user's internet affiliation.

Column: InternetAffiliationCode

Description

This is the user's internet affiliation.

Column: InternetAffiliationDesc

Description

This column contains a description for an internet affiliation code.

Column: lastName

Description

This column holds the user's last name.

Column: mailAddrCity

Description

This is the user's mailing city address.

Column: mailAddrCountry

Description

This is the user's mailing country address.

Column: mailAddrFax

Description

This is the user's contact fax number.

Column: mailAddrPhone

Description

This is the user's contact phone number.

Column: mailAddrState

Description

This is the user's mailing state address.

Column: mailAddrStreet1

Description

This is the user's mail street address.

Column: mailAddrStreet2

Description

This is the user's mail street address. Used only if address street length is longer than what can be accommodated in mailAddrStreet1.

Column: mailAddrZip

Description

This is the user's mail zip code address.

Column: MediaFormatCode

Description

This is the type of media format.

Column: MediaFormatDesc

Description

This is the description of the media format.

Column: mediaType

Description

This column describes the media type of request distribution.

Column: MediaTypeCode

Description

This column identifies the media type of request distribution.

Column: MediaTypeDesc

Description

This is the description of a media type.

Column: middleInit

Description

This column holds the user's middle name.

Column: motherMaidenName

Description

This is the user's mother's maiden name, for security reasons.

Column: nasaUser

Description

This field identifies whether a user works for NASA.

Valid Values:

Y=Yes
N=No

Column: numBytes

Description

This column contains the number of bytes of a request.

Column: numFiles

Description

This column contains the number of files that fill a request.

Column: numGranule

Description

This column contains the number of granules that fill a request.

Column: orderDesc

Description

This column holds a description of the user's order.

Column: orderDistFormat

Description

This column holds the media format of the user's order.

Valid Values:

tar

Column: orderGranule

Description

This column contains the number of granules that fill an order.

Column: orderHomeDAAC

Description

This column is pass from EcAcOrder, this is the home DAAC where the order was placed (same values as daac_short).

Column: orderId

Description

This column is pass from the EcAcOrder table and identifies an order.

Column: orderMedia

Description

This column holds the media type of the user's order.

Column: orderPriority

Description

This column holds the priority of the user's order.

Column: orderSize

Description

This column holds the size in bytes of the user's order.

Column: orderSource

Description

This column holds the where the source of the order.

Column: orderStatus

Description

This column holds the current status of an order.

Valid Values:

Pending

Operator Intervention
Staging
Transferring
Not Found
Waiting for Shipment
Shipped
Aborted
Canceled
Terminated
Subsetting
Prep. for Distribution
SDSRV Staging

Column: organization

Description

This is the user's organization.

Column: parentId

Description

A request can be broken into subrequests, and this column holds the ID for that request. This is a B1 issue.

Column: PriorityCode

Description

Defines a list of possible priority values.

Column: PriorityDesc

Description

This is the description of a request priority code.

Column: privilegeLevel

Description

This column contains the highest priority level a user can give his or her order.

Column: projectName

Description

This is the user's project name.

Column: receiveDateTime

Description

This attribute holds the time the order and/or request was submitted (i.e., created) to the SDSRV, set by the VO Gateway when it created the EcAcRequest.

Column: request_id

Description

This column holds the identifier for a request.

Column: request_type

Description

This column identifies the type of request received or the type of request to be triggered by a subscription (e.g., "Notification ftp-pull").

Column: requestDesc

Description

This column holds the request's description.

Column: requestDistFormat

Description

This column holds the distribution media format.

Column: requestHomeDAAC

Description

This column holds the home DAAC where the request was placed (same values as daac_short).

Column: requestId

Description

The request identifier is passed from the EcAcRequest.

Column: requestPriority

Description

This column holds the user's request priority.

Column: requestStatus

Description

This column holds the user's request status (same values as orderStatus).

Column: researchField

Description

This is the research field available in the system.

Column: ResearchFieldCode

Description

This is the research field available in the system.

Column: ResearchFieldDesc

Description

This is the research field description.

Column: roleID

Description

The column contains an operator's role.

Column: schemaVersionId

Description

This is the version of the database.

Column: shipAddrCity

Description

This is the user's city address to where the request will be shipped.

Column: shipAddrCountry

Description

This is the user's country address to where the request will be shipped.

Column: shipAddrFax

Description

This is the user's fax number to where the request will be shipped.

Column: shipAddrPhone

Description

This is the user's phone address to where the request will be shipped.

Column: shipAddrState

Description

This is the user's state address to where the request will be shipped.

Column: shipAddrStreet1

Description

This is the user's street address to where the request will be shipped.

Column: shipAddrStreet2

Description

This is the user's street address to where the request will be shipped. Used only when street address length is longer than what can be accommodated in shipAddrStreet1

Column: shipAddrZip

Description

This is the user's zip code address to where the request will be shipped.

Column: shipDateTime

Description

This column holds the time the last request for the order was shipped, this time is set by MSS when propagating request status to the order.

Column: startTime

Description

This column holds the time set by DDIST to the first time DDIST started to process the request, i.e., start the staging of its data, and the request status changed to "staging".

Column: StatusCode

Description

This is the status of a request (same values as orderStatus).

Column: StatusDesc

Description

This is the request status code's description.

Column: stop_time

Description

This is the time processing of a request ended.

Column: tapeFormat

Description

This column holds the format of the tape for the request.

Column: telNum

Description

This is the user's telephone number.

Column: timeOfLastUpdate

Description

This column holds the time of the last order or request update.

Column: title

Description

This is the title of a user. (i.e., Dr.)

Valid Values:

Dr	Doctor
Mr	Mister
Ms	Miss/Mrs.
Miss	Miss
Mrs	Mrs.
Rev	Reverend
Sr	Senior

Column: userid

Description

MsAcUsrProfile: This column uniquely identifies a registered user.

All other tables: A registered user id, a guest userid, or an application id.

3.1.4 Column Domains

Domains specify the ranges of values allowed for a given table column. Sybase supports the definition of specific domains to further limit the format of data for a given column. Sybase domains are, in effect, user-defined data types. There are no domains defined for the MSS databases.

3.1.5 Rules

Sybase supports the definitions of rules. Rules provide a means for enforcing domain constraints on a given column. All rules defined in Sybase for the MSS database are described herein.

There are no rules defined in the MSS databases.

3.1.6 Defaults

Defaults are used to supply a value for a column when one is not defined at insert time. All defaults defined in Sybase in the MSS database are described herein.

There are no defaults defined in the MSS databases.

3.1.7 Views

Sybase allows the definition of views as a means of limiting an application or users access to data in a table or tables. Views create a logical table from columns found in one or more tables. There are no views defined for MSS.

3.1.8 Integrity Constraints

Sybase allows the enforcement of referential integrity via the use of declarative integrity constraints. Integrity constraints allow the SQL server to enforce primary and foreign key integrity checks. Sybase 11 is only ANSI-92 compliant, however, therefore its constraints support “restrict-only” operations. This means that a row can not be deleted or updated if there are rows in other tables having a foreign key dependency on that row. Cascade delete and update operations can not be performed if a declarative constraint has been used. There are no declarative integrity constraints defined in the MSS database.

3.1.9 Triggers

Sybase supports the enforcement of business policy via the use of triggers. A trigger is best defined as set of activities or checks that should be performed automatically whenever a row is

inserted, updated, or deleted from a given table. Sybase allows the definition of insert, update, and delete trigger per table. A listing of each of the triggers in the MSS database is given here. A brief definition of each of these triggers follows.

Table 3-20. Trigger Listing

Table	Trigger	Description
EcAcRequest	TrigInsEcAcRequest	Insert EcAcRequest
EcAcRequest	TrigUpdEcAcRequest	Update EcAcRequest
EcAcRequest	TrigDelEcAcRequest	Delete EcAcRequest
MsAcUsrProfile	TrigInsUpdMsAcUsrProfile	Insert/Update MsAcUsrProfile
MsAcUsrProfile	TrigDelMsAcUsrProfile	Delete MsAcUsrProfile

3.1.9.1 Trigger: TrigInsEcAcRequest

Trigger Code

```

CREATE TRIGGER TrigInsEcAcRequest
ON EcAcRequest
FOR INSERT as
begin

/* commented out until drop5..
*/
** Return if no rows affected.
*/
if @@rowcount = 0
    return

declare @retval int, @err int
declare @usertype varchar(20)
declare @username varchar(30)
declare @dbname varchar(30)

select @username = suser_name(), @dbname = db_name()

exec @retval = RSMaintUserCheck @username, @dbname, @usertype output
select @err = @@error

if (@err != 0 or @retval != 0)
begin
    raiserror 25101 "Error checking for Rep Server maintenance user!"
    return
end

if @usertype = "Rep Server maint"
begin
    return

```

```
end
```

```
/*
** Override the requestHomeDAAC value inserted into this table.
** This will ensure that LOCAL users only insert LOCAL daacs
** into the requestHomeDAAC field. replication maintenance users
** inserting into this table will not get this far in the trigger.
** Therefore, replication maintenance users will always be able to
** insert rows into this table.
*
```

```
UPDATE EcAcRequest
set EcAcRequest.requestHomeDAAC =
    ( select daac_short from L_LOCAL_DAAC )
from EcAcRequest, inserted
where EcAcRequest.requestId = inserted.requestId
```

```
declare @insert_cnt int
declare @req_user_id varchar(30)
declare @req_time datetime
declare @req_time_year int
declare @req_time_mon int
declare @req_time_day int
declare @req_time_hour int
declare @req_time_min int

declare @req_time_yeardc varchar(4)
declare @req_time_mondc varchar(4)
declare @req_time_daydc varchar(4)
declare @req_time_hourc varchar(4)
declare @req_time_minc varchar(4)
```

```
declare @time_to_match datetime
declare @email varchar(64)
declare @esdt varchar(20)
declare @media char(30)
declare @granules int
declare @files int
declare @bytes int
declare @collection varchar(30)
declare @requestId varchar(10)
```

```
select @requestId = min(requestId)
      from inserted
```

```
while @requestId != null
BEGIN
```

```
select @email = eMailAddr,
       @collection = ESDT_Id,
       @media = mediaType,
```

```

        @req_time = receiveDateTime
from inserted
where requestId = @requestId

select @req_user_id = Ord.userId
from inserted ins, EcAcOrder Ord
where ins.orderId = Ord.orderId
and ins.requestId = @requestId

select @req_time_year = datepart(yy,@req_time)
select @req_time_mon = datepart(mm,@req_time)
select @req_time_day = datepart(dd,@req_time)
select @req_time_hour = datepart(hh,@req_time)
select @req_time_min = datepart(mi,@req_time)

if (@req_time_min > 30)
    select @req_time_min = 30
else
    select @req_time_min = 0

select @req_time_yearc = convert(varchar(4), @req_time_year)
select @req_time_monc = convert(varchar(4), @req_time_mon)
select @req_time_dayc = convert(varchar(4), @req_time_day)
select @req_time_hourc = convert(varchar(4), @req_time_hour)
select @req_time_minc = convert(varchar(4), @req_time_min)

select @time_to_match =
convert(datetime, (@req_time_monc + "-" + @req_time_dayc + "-" + @req_time_yearc +
" " + @req_time_hourc + ":" + @req_time_minc))

/* Define the "new" record as insert for processing of USER_DAILY_ACCESS_P      */
/* "Insert" defined as this requestid exists in the EcAcRequest table before this record */
/*   was inserted                                         */ 
/* "Update" defined as all other cases                                     */
/* */

/* Step 2:                                         */
/* See if there is a corresponding record in the USER_DAILY_ACCESS_P for this  */
/* User ID and converted time period                                         */
/* */

IF not exists
( select 1
  from USER_DAILY_ACCESS_P
  where @req_user_id = USER_DAILY_ACCESS_P.user_id
    and @time_to_match = USER_DAILY_ACCESS_P.time_period )
BEGIN

```

```

/* Begin insert of new record into USER_DAILY_ACCESS_P */
/* Increment the corresponding count */

if @media = "ftp"
    INSERT USER_DAILY_ACCESS_P
        (user_id, e_mail, remote_host, collection_name, time_period,
         ftp_order_count, ftp_granules, ftp_files, ftp_bytes)
/*
** Modified 8/8/97 by peter macharrie
** receive_date_time field was populated with time
** stamp value that was not on a half hour increment
** original statement
** VALUES (@req_user_id, @email, "      ", @collection, @req_time,
**          1, 0, 0, 0 )
*/
VALUES (@req_user_id, @email, "      ", @collection, @time_to_match,
       1, 0, 0, 0 )

else
    INSERT USER_DAILY_ACCESS_P
        (user_id, e_mail, remote_host, collection_name, time_period,
         media_order_count, media_granules, media_files, media_bytes)
/*
** Modified 8/8/97 by peter macharrie
** receive_date_time field was populated with time
** stamp value that was not on a half hour increment
** original statement
** VALUES (@req_user_id, @email, "      ", @collection, @req_time,
**          1, 0, 0, 0 )
*/
VALUES (@req_user_id, @email, "      ", @collection, @time_to_match,
       1, 0, 0, 0 )
END

```

```

/* BEGIN "update" processing
/* Step 1      : Get selected fields from the inserted record */

select @granules = isnull(numGranule,0),
       @files = isnull(numFiles,0),
       @bytes = isnull(numBytes,0),
       @esdt = isnull(ESDT_Id," ")
from inserted
where requestId = @requestId

if @media = "ftp"
    UPDATE USER_DAILY_ACCESS_P
        SET USER_DAILY_ACCESS_P.ftp_order_count = USER_DAILY_ACCESS_P.ftp_order_count + 1,
            USER_DAILY_ACCESS_P.ftp_granules = USER_DAILY_ACCESS_P.ftp_granules + @granules,
            USER_DAILY_ACCESS_P.ftp_files = USER_DAILY_ACCESS_P.ftp_files + @files,

```

```

        USER_DAILY_ACCESS_P.ftp_bytes = USER_DAILY_ACCESS_P.ftp_bytes + @bytes
    FROM USER_DAILY_ACCESS_P
    WHERE USER_DAILY_ACCESS_P.user_id = @req_user_id
        and USER_DAILY_ACCESS_P.time_period = @time_to_match

else
    UPDATE USER_DAILY_ACCESS_P
        SET USER_DAILY_ACCESS_P.media_order_count = USER_DAILY_ACCESS_P.media_order_count + 1,
            USER_DAILY_ACCESS_P.media_granules = USER_DAILY_ACCESS_P.media_granules + @granules,
            USER_DAILY_ACCESS_P.media_files = USER_DAILY_ACCESS_P.media_files + @files,
            USER_DAILY_ACCESS_P.media_bytes = USER_DAILY_ACCESS_P.media_bytes + @bytes
    FROM USER_DAILY_ACCESS_P
    WHERE USER_DAILY_ACCESS_P.user_id = @req_user_id
        and USER_DAILY_ACCESS_P.time_period = @time_to_match

```

** End of commented out lines */

```

select @requestId = min(requestId)
    from inserted
    where requestId > @requestId
END

```

```

/*
** If a LOCAL user, update the aggregates in the EcAcOrder table.
** Replicate maintenance users will replicate the updates instead
** of recalculating them through the following trigger code.
*/

```

```

declare @requestHomeDAAC varchar(10)
declare @orderId varchar(10)
declare @orderHomeDAAC varchar(10)
declare @rows int

```

/* update EcAcOrder table aggregate columns */

```

select @requestId = min(requestId)
    from inserted

while @requestId != null
begin
    select @requestHomeDAAC = min(requestHomeDAAC)
        from inserted
        where requestId = @requestId

    while @requestHomeDAAC != null
    begin

        select @orderId = orderId,
            @orderHomeDAAC = orderHomeDAAC
            from inserted
            where requestId = @requestId

```

```

and requestHomeDAAC = @requestHomeDAAC

update EcAcOrder
    set orderGranule = (select sum(numGranule)
        from EcAcRequest
        where orderId = @orderId
            and orderHomeDAAC = @orderHomeDAAC),
    orderSize = (select sum(numBytes)
        from EcAcRequest
        where orderId = @orderId
            and orderHomeDAAC = @orderHomeDAAC)
    from EcAcOrder e1
    where e1.orderId = @orderId
        and e1.orderHomeDAAC = @orderHomeDAAC

select @rows = @@rowcount, @err = @@error

if @err != 0
begin
    print "Error updating EcAcOrder table aggregates."
    rollback transaction
    return
end

if @rows > 1
begin
    print "EcAcRequest insertedTrigger: orderId is not unique."
    rollback transaction
    return
end

if @rows = 0
begin
    print "EcAcRequest inserted Trigger: join to EcAcOrder table failed."
    rollback transaction
    return
end

select @requestHomeDAAC = min(requestHomeDAAC)
    from inserted
    where requestId = @requestId
        and requestHomeDAAC > @requestHomeDAAC
end

select @requestId = min(requestId)
    from inserted
    where requestId > @requestId
end
end
go

```

3.1.9.2 Trigger: TrigUpdEcAcRequest

Trigger Code

```
create trigger TrigUpdEcAcRequest
on EcAcRequest
for update as
begin

/* commented out lines until drop5...
*/

$$** \text{ Return if no rows affected.}$$


$$*/$$

IF @@rowcount = 0
    return

declare @retval int, @err int
declare @usertype varchar(20)
declare @username varchar(30)
declare @dbname varchar(30)

select @username = suser_name(), @dbname = db_name()

exec @retval = RSMaintUserCheck @username, @dbname, @usertype output
select @err = @@error
if (@err != 0 or @retval != 0)
begin
    raiserror 25101 "Error checking for Rep Server maintenance user!"
    return
end

if @usertype = "Rep Server maint"
begin
    return
end

/*
** Override the requestHomeDAAC value inserted into this table.
** This will ensure that LOCAL users only insert LOCAL daacs
** into the requestHomeDAAC field. replication maintenance users
** inserting into this table will not get this far in the trigger.
** Therefore, replication maintenance users will always be able to
** insert rows into this table.
*/
UPDATE EcAcRequest
    set EcAcRequest.requestHomeDAAC =
        ( select daac_short from L_LOCAL_DAAC )
    from EcAcRequest, inserted
    where EcAcRequest.requestId = inserted.requestId
```

```

declare @insert_cnt int
declare @req_user_id varchar(30)
declare @req_time datetime
declare @req_time_year int
declare @req_time_mon int
declare @req_time_day int
declare @req_time_hour int
declare @req_time_min int

declare @req_time_yearc varchar(4)
declare @req_time_monc varchar(4)
declare @req_time_dayc varchar(4)
declare @req_time_hourc varchar(4)
declare @req_time_minc varchar(4)

declare @time_to_match datetime
declare @email varchar(64)
declare @esdt varchar(20)
declare @media char(30)
declare @granules int
declare @files int
declare @bytes int
declare @collection varchar(30)

declare @requestId varchar(10)

select @requestId = min(requestId)
from inserted

while @requestId != null
BEGIN

    select @email = eMailAddr,
           @collection = ESDT_Id,
           @media = mediaType,
           @req_time = receiveDateTime
    from inserted
    where requestId = @requestId

    select @req_user_id = Ord.userId
    from inserted ins, EcAcOrder Ord
    where ins.orderId = Ord.orderId
        and ins.requestId = @requestId

    select @req_time_year = datepart(yy,@req_time)
    select @req_time_mon = datepart(mm,@req_time)
    select @req_time_day = datepart(dd,@req_time)
    select @req_time_hour = datepart(hh,@req_time)
    select @req_time_min = datepart(mi,@req_time)

    if (@req_time_min > 30)

```

```

select @req_time_min = 30
else
    select @req_time_min = 0

select @req_time_yearc = convert(varchar(4), @req_time_year)
select @req_time_monc = convert(varchar(4), @req_time_mon)
select @req_time_dayc = convert(varchar(4), @req_time_day)
select @req_time_hourc = convert(varchar(4), @req_time_hour)
select @req_time_minc = convert(varchar(4), @req_time_min)

select @time_to_match =
convert(datetime, (@req_time_monc + "-" + @req_time_dayc + "-" + @req_time_yearc +
" " + @req_time_hourc + ":" + @req_time_minc))

/* Define the "new" record as insert for processing of USER_DAILY_ACCESS_P      */
/* "Insert" defined as this requestid exists in the EcAcRequest table before this record */
/*   was inserted                                         */
/* "Update" defined as all other cases                                     */

/* Step 2:                                         */
/* See if there is a corresponding record in the USER_DAILY_ACCESS_P for this  */
/* User ID and converted time period                                         */

IF not exists
( select 1
  from USER_DAILY_ACCESS_P
  where @req_user_id = USER_DAILY_ACCESS_P.user_id
    and @time_to_match = USER_DAILY_ACCESS_P.time_period )
BEGIN

/* Begin insert of new record into USER_DAILY_ACCESS_P */
/* Increment the corresponding count                   */

if @media = "ftp"
  INSERT USER_DAILY_ACCESS_P
    (user_id, e_mail, remote_host, collection_name, time_period,
     ftp_order_count, ftp_granules, ftp_files, ftp_bytes)
/*
** Modified 8/8/97 by peter macharrie
** receieve_date_time field was populated with time
** stamp value that was not on a half hour increment
** original statement
** VALUES (@req_user_id, @email, "      ", @collection, @req_time,
**         1, 0, 0, 0 )
*/

```

```

VALUES (@req_user_id, @email, "      ", @collection, @time_to_match,
1, 0, 0, 0 )

else
  INSERT USER_DAILY_ACCESS_P
    (user_id, e_mail, remote_host, collection_name, time_period,
     media_order_count, media_granules, media_files, media_bytes)

/*
** Modified 8/8/97 by peter macharrie
** receive_date_time field was populated with time
** stamp value that was not on a half hour increment
** original statement
** VALUES (@req_user_id, @email, "      ", @collection, @req_time,
**        1, 0, 0, 0 )
*/
VALUES (@req_user_id, @email, "      ", @collection, @time_to_match,
1, 0, 0, 0 )
END

```

```

/* BEGIN "update" processing          */
/* Step 1   : Get selected fields from the inserted record */

select @granules = isnull(numGranule,0),
       @files = isnull(numFiles,0),
       @bytes = isnull(numBytes,0),
       @esdt = isnull(ESDT_Id," ")
from inserted
where requestId = @requestId

if @media = "ftp"
  UPDATE USER_DAILY_ACCESS_P
    SET USER_DAILY_ACCESS_P.ftp_order_count = USER_DAILY_ACCESS_P.ftp_order_count + 1,
        USER_DAILY_ACCESS_P.ftp_granules = USER_DAILY_ACCESS_P.ftp_granules + @granules,
        USER_DAILY_ACCESS_P.ftp_files = USER_DAILY_ACCESS_P.ftp_files + @files,
        USER_DAILY_ACCESS_P.ftp_bytes = USER_DAILY_ACCESS_P.ftp_bytes + @bytes
  FROM USER_DAILY_ACCESS_P
  WHERE USER_DAILY_ACCESS_P.user_id = @req_user_id
    and USER_DAILY_ACCESS_P.time_period = @time_to_match

else
  UPDATE USER_DAILY_ACCESS_P
    SET USER_DAILY_ACCESS_P.media_order_count = USER_DAILY_ACCESS_P.media_order_count + 1,
        USER_DAILY_ACCESS_P.media_granules = USER_DAILY_ACCESS_P.media_granules + @granules,
        USER_DAILY_ACCESS_P.media_files = USER_DAILY_ACCESS_P.media_files + @files,
        USER_DAILY_ACCESS_P.media_bytes = USER_DAILY_ACCESS_P.media_bytes + @bytes
  FROM USER_DAILY_ACCESS_P
  WHERE USER_DAILY_ACCESS_P.user_id = @req_user_id
    and USER_DAILY_ACCESS_P.time_period = @time_to_match

```

```

select @requestId = min(requestId)
from inserted
where requestId > @requestId
END

** End of commned out lines */

/*
** If a LOCAL user, update the aggregates in the EcAcOrder table.
** Replicate maintenance users will replicate the updates instead
** of recalculating them through the following trigger code.
*/

/* if numGranule and numBytes were not updated, then return */

IF not update(numGranule) and
not update(numBytes) and
not update(orderId) and
not update(orderHomeDAAC)
return

/* update EcAcOrder table aggregate columns */

declare @requestHomeDAAC varchar(10)
declare @orderId varchar(10)
declare @orderHomeDAAC varchar(10)
declare @rows int

if update (requestId) or
update (requestHomeDAAC)

begin
select @requestId = min(requestId)
from deleted

while @requestId != null
begin

select @requestHomeDAAC = min(requestHomeDAAC)
from deleted
where requestId = @requestId

while @requestHomeDAAC != null
begin

select @orderId = orderId,
@orderHomeDAAC = orderHomeDAAC
from deleted
where requestId = @requestId
and requestHomeDAAC = @requestHomeDAAC

```

```

update EcAcOrder
    set orderGranule = (select sum(numGranule)
        from EcAcRequest
        where orderId = @orderId
        and orderHomeDAAC = @orderHomeDAAC),
    orderSize = (select sum(numBytes)
        from EcAcRequest
        where orderId = @orderId
        and orderHomeDAAC = @orderHomeDAAC)
from EcAcOrder e1
where e1.orderId = @orderId
    and e1.orderHomeDAAC = @orderHomeDAAC

select @rows = @@rowcount, @err = @@error
if @err != 0
begin
    print "Error updating EcAcOrder table aggregates."
    rollback transaction
    return
end

if @rows > 1
begin
    print "EcAcRequest delete Trigger: orderId is not unique."
    rollback transaction
    return
end

if @rows = 0
begin
    print "EcAcRequest delete Trigger: join to EcAcOrder table failed."
    rollback transaction
    return
end

select @requestHomeDAAC = min(requestHomeDAAC)
from deleted
where requestId = @requestId
    and requestHomeDAAC > @requestHomeDAAC
end

select @requestId = min(requestId)
    from deleted
    where requestId > @requestId
end
end

select @requestId = min(requestId)
    from inserted

while @requestId != null

```

```

begin
select @requestHomeDAAC = min(requestHomeDAAC)
from inserted
where requestId = @requestId

while @requestHomeDAAC != null
begin

select @orderId = orderId,
       @orderHomeDAAC = orderHomeDAAC
from inserted
where requestId = @requestId
and requestHomeDAAC = @requestHomeDAAC

update EcAcOrder
set orderGranule = (select sum(numGranule)
                     from EcAcRequest
                     where orderId = @orderId
                     and orderHomeDAAC = @orderHomeDAAC),
orderSize = (select sum(numBytes)
             from EcAcRequest
             where orderId = @orderId
             and orderHomeDAAC = @orderHomeDAAC)
from EcAcOrder e1
where e1.orderId = @orderId
and e1.orderHomeDAAC = @orderHomeDAAC

select @rows = @@rowcount, @err = @@error

if @err != 0
begin
    print "Error updating EcAcOrder table aggregates."
    rollback transaction
    return
end

if @rows > 1
begin
    print "EcAcRequest inserted Trigger: orderId is not unique."
    rollback transaction
    return
end

if @rows = 0
begin
    print "EcAcRequest inserted Trigger: join to EcAcOrder table failed."
    rollback transaction
    return
end

select @requestHomeDAAC = min(requestHomeDAAC)
from inserted

```

```

    where requestId = @requestId
        and requestHomeDAAC > @requestHomeDAAC
    end

    select @requestId = min(requestId)
        from inserted
        where requestId > @requestId
    end
end
go

```

3.1.9.3 Trigger: TrigDelEcAcRequest

Trigger Code

```

create trigger TrigDelEcAcRequest
on EcAcRequest
for
delete
as

/*
** Return if no rows affected.
*/
IF @@rowcount = 0
    return

/*
** Check to see if this insert update is being made be replication server
** maintenance user. If so, don't bother to do any other checking.
*/
declare @retval int, @err int
declare @usertype varchar(20)
declare @username varchar(30)
declare @dbname varchar(30)

select @username = suser_name(), @dbname = db_name()

exec @retval = RSMaintUserCheck @username, @dbname, @usertype output
select @err = @@error

if (@err != 0 or @retval != 0)
begin
    raiserror 25101 "Error checking for Rep Server maintenance user!"
    return
end

if @usertype = "Rep Server maint"
begin
    return
end

```

```

declare @requestId varchar(10),
        @requestHomeDAAC varchar(10),
        @orderId varchar(10),
        @orderHomeDAAC varchar(10),
        @rows int

/* update EcAcOrder table aggregate columns */

select @requestId = min(requestId)
      from deleted

while @requestId != null
begin

select @requestHomeDAAC = min(requestHomeDAAC)
      from deleted
     where requestId = @requestId

while @requestHomeDAAC != null
begin

select @orderId = orderId,
       @orderHomeDAAC = orderHomeDAAC
      from deleted
     where requestId = @requestId
       and requestHomeDAAC = @requestHomeDAAC

update EcAcOrder
  set orderGranule = (select sum(numGranule)
                        from EcAcRequest
                       where orderId = @orderId
                         and orderHomeDAAC = @orderHomeDAAC),
      orderSize = (select sum(numBytes)
                   from EcAcRequest
                  where orderId = @orderId
                    and orderHomeDAAC = @orderHomeDAAC)
  from EcAcOrder e1
 where e1.orderId = @orderId
   and e1.orderHomeDAAC = @orderHomeDAAC

select @rows = @@rowcount, @err = @@error

if @err != 0
begin
  print "Error updating EcAcOrder table aggregates."
  rollback transaction
  return
end

if @rows > 1
begin

```

```

        print "EcAcRequest delete Trigger: orderId is not unique."
        rollback transaction
        return
    end

    if @rows = 0
    begin
        print "EcAcRequest delete Trigger: join to EcAcOrder table failed."
        rollback transaction
        return
    end

    select @requestHomeDAAC = min(requestHomeDAAC)
    from deleted
    where requestId = @requestId
        and requestHomeDAAC > @requestHomeDAAC
end

select @requestId = min(requestId)
from deleted
where requestId > @requestId
end
go

```

3.9.1.4 Trigger: TrigInsUpdMsAcUsrProfile

Trigger Code

```

CREATE TRIGGER TrigInsUpdMsAcUsrProfile
ON MsAcUsrProfile
FOR INSERT, UPDATE as
begin

/*
** Return if no rows affected.
*/
if @@rowcount = 0
    return

/*
** Make sure the homedaac is always for this particular daac
** so that this will not interfere with the replication to smc
*/
declare @retval int, @err int
declare @usertype varchar(20)
declare @username varchar(30)
declare @dbname varchar(30)

select @username = suser_name(), @dbname = db_name()

exec @retval = RSMaintUserCheck @username, @dbname, @usertype output

```

```

select @err = @@error
if (@err != 0 or @retval != 0)
begin
    raiserror 25101 "Error checking for Rep Server maintenance user!"
    return
end

if @usertype = "Rep Server maint"
begin
    return
end

if (select count (*) from inserted
    where homeDAAC not in
        (select daac_short from L_LOCAL_DAAC)) > 0
begin
    declare @daac varchar(10)
    select @daac=daac_short from L_LOCAL_DAAC
    raiserror 25100 "Daac name must be %1!", @daac
    rollback transaction
    return
end
else
begin

if exists (select * from inserted where eMailAddr is not null)
/* Can only populate where the insert record is valued on eMailAddr */

BEGIN

/* US Government */

/* Case 1: the address ends in .gov or .mil */

if exists ( select right(eMailAddr,4) from inserted
            where right (eMailAddr,4) in ("."gov","."mil" ) )
begin
    UPDATE MsAcUsrProfile
    set MsAcUsrProfile.internetAffiliation = "US Government"
    from MsAcUsrProfile, inserted
    where MsAcUsrProfile.userId = inserted.userId and
          right(inserted.eMailAddr,4) in ("."gov","."mil" )
    return
end

/* Case 2: the address ends in .us and has .gov. or .mil in string */

if exists ( select eMailAddr from inserted
            where right (eMailAddr,3) = '.us' and
                  ( charindex(".gov.",eMailAddr) > 0 or
                    charindex(".mil.",eMailAddr) > 0 ) )
begin
    UPDATE MsAcUsrProfile

```

```

set MsAcUsrProfile.internetAffiliation = "US Government"
from MsAcUsrProfile, inserted
where MsAcUsrProfile.userId = inserted.userId and
      right (inserted.eMailAddr,3) = '.us' and
      ( charindex(".gov.",inserted.eMailAddr) > 0 or
        charindex(".mil.",inserted.eMailAddr) > 0 )
return
end

/* Educational */

/* Case 3: the address ends in .edu or .k12 */

if exists ( select right(eMailAddr,4) from inserted
            where right (eMailAddr,4) in ("edu",".k12" ) )
begin
    UPDATE MsAcUsrProfile
    set MsAcUsrProfile.internetAffiliation = "Educational"
    from MsAcUsrProfile, inserted
    where MsAcUsrProfile.userId = inserted.userId and
          right(inserted.eMailAddr,4) in ("edu",".k12" )
    return
end

/* Case 4: the address ends in .us and has .edu. or .k12 in string */

if exists ( select eMailAddr from inserted
            where right (eMailAddr,3) = '.us' and
                  ( charindex(".edu.",eMailAddr) > 0 or
                    charindex(".k12.",eMailAddr) > 0 ) )
begin
    UPDATE MsAcUsrProfile
    set MsAcUsrProfile.internetAffiliation = "Educational"
    from MsAcUsrProfile, inserted
    where MsAcUsrProfile.userId = inserted.userId and
          ( charindex(".edu.",inserted.eMailAddr) > 0 or
            charindex(".k12.",inserted.eMailAddr) > 0 )
    return
end

/* Commercial */

/* Case 5: the address ends in .com or .net */

if exists ( select right(eMailAddr,4) from inserted
            where right (eMailAddr,4) in ("com",".net" ) )
begin
    UPDATE MsAcUsrProfile
    set MsAcUsrProfile.internetAffiliation = "Commercial"
    from MsAcUsrProfile, inserted
    where MsAcUsrProfile.userId = inserted.userId and
          right (inserted.eMailAddr,4) in ("com",".net" )
    return

```

```

end

/* Case 6: the address ends in .us and has .com. or .net. in string */

if exists ( select eMailAddr from inserted
            where right (eMailAddr,3) = '.us' and
                  ( charindex(".com.",eMailAddr) > 0 or
                    charindex(".net.",eMailAddr) > 0 ) )
begin
    UPDATE MsAcUsrProfile
    set MsAcUsrProfile.internetAffiliation = "Commercial"
    from MsAcUsrProfile, inserted
    where MsAcUsrProfile.userId = inserted.userId and
          right (inserted.eMailAddr,3) = '.us' and
          ( charindex(".com.", inserted.eMailAddr) > 0 or
            charindex(".net.", inserted.eMailAddr) > 0 )
    return
end

/* Non-Profit */

/* Case 7: the address ends in .org */

if exists ( select right(eMailAddr,4) from inserted
            where right (eMailAddr,4) = ".org" )
begin
    UPDATE MsAcUsrProfile
    set MsAcUsrProfile.internetAffiliation = "Non-Profit"
    from MsAcUsrProfile, inserted
    where MsAcUsrProfile.userId = inserted.userId and
          right(inserted.eMailAddr,4) = ".org"
    return
end

/* Case 8: the address ends in .us and has .com. or .net. in string */

if exists ( select eMailAddr from inserted
            where right (eMailAddr,3) = '.us' and
                  charindex(".org.",eMailAddr) > 0 )
begin
    UPDATE MsAcUsrProfile
    set MsAcUsrProfile.internetAffiliation = "Non-Profit"
    from MsAcUsrProfile, inserted
    where MsAcUsrProfile.userId = inserted.userId and
          right (inserted.eMailAddr,3) = '.us' and
          charindex(".org.",inserted.eMailAddr) > 0
    return
end

/* Other USA */

/* Case 9: the address ends in .us and meets none of the above
   criteria (cases 1-8) */

```

```

if exists ( select eMailAddr from inserted
    where right (eMailAddr,3) = '.us' and
        charindex(".gov.",eMailAddr) = 0 and
        charindex(".mil.",eMailAddr) = 0 and
        charindex(".edu.",eMailAddr) = 0 and
        charindex(".k12.",eMailAddr) = 0 and
        charindex(".com.",eMailAddr) = 0 and
        charindex(".net.",eMailAddr) = 0 and
        charindex(".org.",eMailAddr) = 0 )
begin
    UPDATE MsAcUsrProfile
    set MsAcUsrProfile.internetAffiliation = "Other USA"
    from MsAcUsrProfile, inserted
    where MsAcUsrProfile.userId = inserted.userId and
        right (inserted.eMailAddr,3) = '.us' and
        charindex(".gov.",inserted.eMailAddr) = 0 and
        charindex(".mil.",inserted.eMailAddr) = 0 and
        charindex(".edu.",inserted.eMailAddr) = 0 and
        charindex(".k12.",inserted.eMailAddr) = 0 and
        charindex(".com.",inserted.eMailAddr) = 0 and
        charindex(".net.",inserted.eMailAddr) = 0 and
        charindex(".org.",inserted.eMailAddr) = 0
    return
end

```

/* Case 10: ends in a period + 3 chars */

```

if exists ( select eMailAddr from inserted
    where substring(right(eMailAddr,4),1,1) = '.' and
        substring(right(eMailAddr,2),2,1) like '[a-z]' and
        substring(right(eMailAddr,2),3,1) like '[a-z]' and
        substring(right(eMailAddr,2),4,1) like '[a-z]' )
begin
    UPDATE MsAcUsrProfile
    set MsAcUsrProfile.internetAffiliation = "Other USA"
    from MsAcUsrProfile, inserted
    where MsAcUsrProfile.userId = inserted.userId and
        substring(right(inserted.eMailAddr,4),1,1) = '.' and
        substring(right(inserted.eMailAddr,2),2,1) like '[a-z]' and
        substring(right(inserted.eMailAddr,2),3,1) like '[a-z]' and
        substring(right(inserted.eMailAddr,2),4,1) like '[a-z]'
    return
end

```

/* Foreign */

```

/* Case 11: ending in decimal plus two chars only (not us) */

if exists ( select inserted.eMailAddr from inserted
    where substring(right(inserted.eMailAddr,3),1,1) = '.' and
        substring(right(inserted.eMailAddr,2),2,2) <> 'us' )
begin

```

```

UPDATE MsAcUsrProfile
set MsAcUsrProfile.internetAffiliation = "Foreign"
from MsAcUsrProfile, inserted
where MsAcUsrProfile.userId = inserted.userId and
      substring(right(inserted.eMailAddr,3),1,1) = '.' and
      substring(right(inserted.eMailAddr,2),2,2) <> 'us'
return
end
end
END

/* end of homedaac ok */
end
go

```

3.1.9.5 Trigger: TrigDelMsAcUsrProfile

Trigger Code

```

CREATE TRIGGER TrigDelMsAcUsrProfile
ON MsAcUsrProfile
FOR DELETE as
begin

/*
** Return if no rows affected.
*/
if @@rowcount = 0
    return

/*
** Make sure the homedaac is always for this particular daac
** so that this will not interfere with the replication to smc
*/
declare @retval int, @err int
declare @usertype varchar(20)
declare @username varchar(30)
declare @dbname varchar(30)

select @username = suser_name(), @dbname = db_name()

exec @retval = RSMaintUserCheck @username, @dbname, @usertype output
select @err = @@error
if (@err != 0 or @retval != 0)
begin
    raiserror 25101 "Error checking for Rep Server maintenance user!"
    return
end

if @usertype = "Rep Server maint"
begin

```

```

return
end

if (select count (*) from deleted
    where homeDAAC not in
        (select daac_short from L_LOCAL_DAAC)) > 0
begin
    declare @daac varchar(10)
    select @daac=daac_short from L_LOCAL_DAAC
    raiserror 25100 "Daac name must be %1!", @daac
    rollback transaction
    return
end

end
go

```

3.1.10 Stored Procedures

Sybase also includes support for business policy via the use of stored procedures. Stored procedures are typically used to capture a set of activities or checks that will be performed on the database repeatedly to enforce business policy and maintain data integrity. Stored procedures are parsed and complied SQL code that reside in the database and may be called by name by an application, trigger or another stored procedure. A listing of each the stored procedures in the MSS database is given here. A brief definition of each of these stored procedures follows.

Table 3-21. Procedure Listing

Name	Description
datawarning	Notifies DBA when a data segment threshold is crossed.
logdump	Dump the log when log segment threshold is crossed.
logwarning	Notify the DBA when log segment approaches capacity threshold.
ProInrementOrderId	Passes back the next order Id to be used as a key for the EcAcOrder.
ProInrementEcAcRequestId	Passes the next request Id to be used as a key for the EcAcRequest.

3.1.10.1 Procedure: datawarning

```

CREATE PROCEDURE datawarning
    @dbname varchar(30),
    @segmentname varchar(30),
    @space_left int,
    @status int
AS
DECLARE @date datetime
    SELECT @date = getdate()
PRINT "DATA/INDEX SEGMENT WARNING: database '%1!', threshold '%2!', segment '%3!', date/time '%4!'",
    @dbname, @space_left, @segmentname, @date

```

go

3.1.10.2 Procedure: logdump

```
CREATE PROCEDURE logdump
    @dbname varchar(30),
    @segmentname varchar(30),
    @space_left int,
    @status int
AS

DECLARE @devname varchar(100),
@before_size int,
@after_size int,
@before_time datetime,
@after_time datetime,
@error int,
@date char(8),
@time char(4)

/* get the time and log size just before the dump */

SELECT @before_time = getdate(), @before_size = reserved_pgs(id, doampg)
FROM sysindexes
WHERE sysindexes.name = "syslogs"
PRINT "LOG DUMP: database '%1!', threshold '%2!'", @dbname, @space_left

/* get current date and time in format suitable for a filename suffix */

SELECT @date = datename(yy,getdate())+
    right('0'+ltrim(convert(varchar(2),
    datepart(mm,getdate()))),2)+
    right('0'+ltrim(convert(varchar(2),
    datepart(dd,getdate()))),2)

SELECT @time = right('0'+ltrim(convert(char(2),
    datepart(hh,getdate()))),2)+
    right('0'+ltrim(convert(char(2),
    datepart(mi,getdate()))),2)

/* dump device is a file in a mode-dependent directory owned by sybase */

SELECT @devname = "/usr/ecs/FRT/COTS/sybase/sybase_dumps/"
    + @dbname + "_tran_dmp." + @date + @time

/* dump the transaction log to the specified device */

DUMP TRANSACTION @dbname TO @devname
SELECT @error = @@error
IF @error != 0
```

```

BEGIN
PRINT "LOG DUMP ERROR: %1!", @error
RETURN @error
END

/* get size of log and time after dump */

SELECT @after_time= getdate(), @after_size =
       reserved_pgs(id, doampg)
FROM sysindexes
WHERE sysindexes.name = "syslogs"

/* print message to error log */

PRINT "LOG DUMPED TO: device '%1!'", @devname
PRINT "LOG DUMP PAGES: Before: '%1!', After '%2!'", 
      @before_size, @after_size
PRINT "LOG DUMP TIME: %1!, %2!",
      @before_time, @after_time
go

```

3.1.10.3 Procedure: logwarning

```

CREATE PROCEDURE logwarning
    @dbname varchar(30),
    @segmentname varchar(30),
    @space_left int,
    @status int
AS
DECLARE @date datetime
    SELECT @date = getdate()
PRINT "LOG WARNING: database '%1!', threshold '%2!', date/time '%3!'",
    @dbname, @space_left, @date
go

```

Procedure 3.1.10.4 : ProcIncrementEcAcRequestId

Description

Code

```
CREATE PROCEDURE ProcIncrementEcAcRequestId
    (@RequestIdString varchar(10) output)
AS
BEGIN
    BEGIN TRAN

    DECLARE @rcount int
    SELECT @rcount = count(*) FROM EcAcRequestId
    IF (@rcount > 1)
        BEGIN
            RAISERROR 93000
            ROLLBACK TRAN
            RETURN 93000
        END
    IF (@rcount = 0)
        INSERT INTO EcAcRequestId VALUES (0)

    UPDATE EcAcRequestId
        SET requestId = requestId + 1

    IF @@error != 0
        BEGIN
            RAISERROR 93000
            ROLLBACK TRAN
            RETURN 93000
        END
    SELECT @RequestIdString = convert(varchar(10),MAX(requestId))
        FROM EcAcRequestId

    IF @@error != 0
        BEGIN
            RAISERROR 93000
            ROLLBACK TRAN
            RETURN 93000
        END

    COMMIT TRAN
END
```

Procedure 3.1.10.5 : ProcIncrementOrderId

Description

Code

```
CREATE PROCEDURE ProcIncrementOrderId
    (@OrderIdString varchar(10) output)
AS
BEGIN
    BEGIN TRAN
    DECLARE @rCount int
    SELECT @rCount = count(*) FROM EcAcOrderId
    IF (@rCount > 1)
        BEGIN
            RAISERROR 93001
            ROLLBACK TRAN
            RETURN 93001
        END
    IF (@rCount = 0)
        INSERT INTO EcAcOrderId VALUES (0)

    UPDATE EcAcOrderId
        SET orderId = orderId + 1

    IF @@error != 0
        BEGIN
            RAISERROR 93001
            ROLLBACK TRAN
            RETURN 93001 rId
        END
    IF @@error != 0
        BEGIN
            RAISERROR 93001
            ROLLBACK TRAN
            RETURN 93001
        END
    COMMIT TRAN
END
END
```

3.2 Flat File Usage

A flat file is an operating system file that is written and subsequently read, generally independent of other files that exist, and usually static in nature. There are cases when the implementation of persistent data is better suited to a flat file than to a database. MSS Subsystem file usage is detailed in this section via file, block, field, and domain definitions.

3.2.1 File Descriptions

A summary listing of the files in the MSS Subsystem is given in Table 3-23 together with a brief description of the file usage. Many different record formats are used in ECS including ODL, HDF, HDF EOS, block, fixed length, variable length, etc.

Table 3-22. Flat File Descriptions (1 of 2)

File Name	File Type	Record Format	File Description
Accountability component files			
MsAcAffiliation.dat	ASCII	Single line records. One field	Contains the list of valid affiliation names for selection of the user's affiliation.
MsAcAsterCategory.dat	ASCII	Single line records. Two fields.	Contains the list of valid Aster DAR user categories for selection of the user's category.
MsAcCountry.dat	ASCII	Single line records. One field	Contains the list of valid country names for selecting a user's country of residence.
MsAcDceGroup.dat	ASCII	Single line records. One field	Contains DCE account group. (e.g. SCIENTIST, ENGINEER)
MsAcDceOrganization.dat	ASCII	Single line records. One field	DCE account organization. (e.g. NASA)
MsAcGateWayType.dat	ASCII	Single line records. One field	User type. (e.g. DAACOPS, GUEST)
MsAcHomeDAAC.dat	ASCII	Single line records. One field	Contains a list of valid DAAC names for selection of the user's home DAAC.
MsAcNasaUser.dat	ASCII	Single line records. One field	Contains Y (yes) and N (no) to indicate NASA user or not.
MsAcPrimaryAreaStudy.dat	ASCII	Single line records. One field	Contains the list of primary study areas for selecting the user's area.
MsAcPrivilegeLevel.dat	ASCII	Single line records. One field	Contains a list of privilege levels for selecting the user's privilege level. (any other insight into this?)
MsAcState.dat	ASCII	Single line records. One field	Contains the list of states in the USA for selecting a user's state of residence.
MsAcTitle.dat	ASCII	Single line records. One field	Contains the list of valid titles for selecting a user's title.
MsAcType.dat	ASCII	Single line records. One field	Contains USA or NONE to indicate US user or not.
Subagent component files			
MsAgEventsHoldingFile	binary	EcAgEvent objects	If subagent is not connected to the deputy agent, the events holding file is used to store events to be processed upon connection to deputy.

Table 3-22. Flat File Descriptions (2 of 2)

File Name	File Type	Record Format	File Description
MsAgBindingVectorFile	binary	MsAgMgmtHandle object	The binding vector file contains binding information to ecs servers such as the uuid and the mode. Whenever a running ecs server is detected by subagent, it adds an entry to the binding vector file. Similarly, when a ecs servers is shutdown or dies for any other reason, its entry is removed from the binding vector. If the subagent is restarted, it uses the information in the binding vector to reconnect with the servers that it was previously monitoring.
MsAgInstanceFile	binary	integers	The instance ID file contains the number that was last assigned to a server by subagent. This number is saved to ensure a unique instance ID across different servers. Whenever subagent needs to assign an instance ID to a server, it reads the value in this file, increments it by one, and then assigns it to the server. The incremented value is written back to the instance ID file.
MsCmActiveModesFile	binary	list of strings	This file contains a list of all the active modes that subagent should discover. The list is maintained by the mode manager GUI.
MsCmAvailableModesFile	binary	list of strings	This file is a master list of all the modes available on all the ecs hosts within a cell. Available modes file is created by each subagent in the cell adding all the installed modes on its host. The purpose of this file is to provide a list of all the modes that can be inserted into the active modes file.

3.2.2 Block Specifications

Table 3-24 identifies the block formats used in MSS files.

Table 3-23. Flat File Block Descriptions (1 of 2)

File Name	Block Name	Block Description
Accountability component file block descriptions		
MsAcAffiliation.dat	(standard)	Single line records; contain 1 field.
MsAcAsterCategory.dat	(standard)	Single line records; contain 2 fields.

Table 3-23. Flat File Block Descriptions (2 of 2)

File Name	Block Name	Block Description
MsAcCountry.dat	(standard)	Single line records; contain 1 field.
MsAcDceGroup.dat	(standard)	Single line records; contain 1 field.
MsAcDceOrganization.dat	(standard)	Single line records; contain 1 field.
MsAcGateWayType.dat	(standard)	Single line records; contain 1 field.
MsAcHomeDAAC.dat	(standard)	Single line records; contain 1 field.
MsAcNasaUser.dat	(standard)	Single line records; contain 1 field.
MsAcPrimaryAreaStudy.dat	(standard)	Single line records; contain 1 field.
MsAcPrivilegeLevel.dat	(standard)	Single line records; contain 1 field.
MsAcState.dat	(standard)	Single line records; contain 1 field.
MsAcTitle.dat	(standard)	Single line records; contain 1 field.
MsAcType.dat	(standard)	Single line records; contain 1 field.

3.2.3 Field Specifications

Brief specifications of the fields present within the MSS Subsystem flat files are contained in Table 3-25. The fields are ordered alphabetically by File Name.

Table 3-24. Flat File Field Specifications

File Name/Block Name	Field Name	Data Type	Field Description
MsAcAffiliation.dat	affiliation	String	Valid affiliation name.
MsAcAsterCategory.dat	Aster category ID	Char 2	Two digit Aster DAR category ID
	Aster category mnemonic	String	Mnemonic corresponding to category ID.
Accountability component field descriptions			
MsAcCountry.dat	country	String	Valid country name.
MsAcDceGroup.dat	DCE group	String	DCE group
MsAcDceOrganization.dat	DCE organization	String	DCE organization
MsAcGateWayType.dat	gateway type	String	Gateway type
MsAcHomeDAAC.dat	home DAAC	String	Valid DAAC name.
MsAcNasaUser.dat	NASA user indication	Char 1	NASA user flag
MsAcPrimaryAreaStudy.dat	primary area of study	String	Valid primary study area
MsAcPrivilegeLevel.dat	privilege level	String	Valid privilege level.
MsAcState.dat	state	String	State of residence.
MsAcTitle.dat	title	String	User's title
MsAcType.dat	type	String	User location type.

3.2.4 Domain Definitions

Domain definitions specify the data type and valid content of fields within a file (e.g., specific values for a limited set of data, ranges of numeric data, units of measure for applicable data). This information is generally used by software to edit incoming data for validity prior to writing or changing data within the file. Use of domain values in updating (adding and changing) records within files preserves the integrity of the data within the file. The domain definitions for the MSS Subsystem are presented in Table 3-26.

Table 3-25. Flat File Domain Definitions

File Name/Block Name	Field Name	Domain Description
Accountability component field domains		
MsAcAffiliation.dat	affiliation	String: K-12 Commercial Government University Other
MsAcAsterCategory.dat	Aster category ID	Integer: 0-99
	Aster category mnemonic	String: (category 0 is not an Aster DAR user) MITI/NASA EOS member IEOS agencies ASTER Science Team Leader US Team Leader ASTER Science Working Groups ASTER Science Team Member AO User Special-Priority Japan user EOS Science Project Office ASTER Science Project (SSSG) ASTER CDS/ESDIS Project ASTER Instrument Team Category 14 Category 15 (through) Category 99

Table 3-25. Flat File Domain Definitions (cont.)

File Name/Block Name	Field Name	Domain Description
MsAcCountry.dat	country	String: Afghanistan Albania Algeria American Samoa Andorra Angola Anguilla Antarctica Antigua and Barbuda Argentina Armenia Aruba Australia Austria Azerbaijan Bahamas Bahrain Bangladesh Barbados Belarus Belgium Belize Benin Bermuda Bhutan Bolivia Bosnia-Hercegovina Botswana Bouvet Island Brazil British Indian Ocean Territory Brunei Darussalam Bulgaria Burkina Faso Burundi Cambodia Cameroon Canada Cape Verde Cayman Islands

Table 3-25. Flat File Domain Definitions (cont.)

File Name/Block Name	Field Name	Domain Description
		Central Africa Republic Chad Chile China Christmas Island Cocos(Keeling) Islands Colombia Comoros Congo Cook Island Costa Rica Cote d'Ivoire(Ivory Coast) Croatia Cuba Cyprus Czech Republic Denmark Djibouti Dominica Dominican Republic East Timor Ecuador Egypt El Salvador Equatorial Guinea Estonia Ethiopia Falkland Islands Faroe Islands Fiji Finland France French Guiana French Polynesia French Southern Territories Gabon Gambia Georgia Germany Ghana Gibraltar Greece

Table 3-25. Flat File Domain Definitions (cont.)

File Name/Block Name	Field Name	Domain Description
		Greenland Grenada Guadeloupe Guam Guatemala Guinea Guinea-Bissau Guyana Haiti heard and McDonald Islands Honduras Hong Kong Hungary Iceland India Indonesia Iran Iraq Ireland Israel Italy Jamaica Japan Jordan Kazakhstan Kenya Kiribati North Korea South Korea Kuwait Kyrgyzstan Lao People's Democratic Republic Latvia Lebanon Lesotho Liberia Libyan Arab Jamahiriya Liechtenstein Lithuania Luxembourg Macau Madagascar

Table 3-25. Flat File Domain Definitions (cont.)

File Name/Block Name	Field Name	Domain Description
		Malawi Malaysia Maldives Mali Malta Marchall Islands Martinique Mauritania Mauritius Mexico Micronesia Moldovia Monaco Mongolia Montserrat Morocco Mozambique Myanmar Namibia Nauru Nepal Netherlands Netherlands Antilles Neutral Zone New Caledonia New Zealand Nicaragua Niger Nigeria Niue Norfolk Island Northern Mariana Islands Norway Oman Pakistan Palau Panama Papua New Guinea Paraguay Peru Phillippines Pitcairn Island

Table 3-25. Flat File Domain Definitions (cont.)

File Name/Block Name	Field Name	Domain Description
		Poland Portugal Puerto Rico Qatar Reunion Island Romania Russian Federation Rwanda St. Helena St. Kitts and Nevis St. Lucia St. Pierre and Miquelon St. Vincent and the Grenadines Samoa San Marino Sao Tome and Principe Saudi Arabia Senegal Seychelles Sierra Leone Singapore Slovak Republic Slovenia Solomon Islands Somalia South Africa Spain Sri Lanka Sudan Suriname Svalbard and Jan Mayen Islands Swaziland Sweden Switzerland Syrian Arab Republic Taiwan Tajikistan Tanzania Thailand Togo Tokelau Tonga

Table 3-25. Flat File Domain Definitions (cont.)

File Name/Block Name	Field Name	Domain Description
		Trinidad and Tobago Tunisia Turkey Turkmenistan Turks and Caicos Islands Tuvalu Uganda Ukraine United Arab Emirates United Kingdom United States Uruguay Uzbekistan Vanuatu Vatican City Venezuela Vietnam Virgin Islands(British) Virgin Islands(U.S.) Wallis and Fortuna Islands Western Sahara Yemen Yugoslavia(former) Zaire Zambia Zimbabwe
MsAcDceGroup.dat	DCE group	???
MsAcDceOrganization.dat	DCE organization	???
MsAcGateWayType.dat	Gateway type	String: DAACOPS ECSDEV V0CERES GUEST
MsAcHomeDAAC.dat	user's home DAAC	String: ASF CSN EDC GSF JPL LAR NSC ORN

Table 3-25. Flat File Domain Definitions (cont.)

File Name/Block Name	Field Name	Domain Description
MsAcNasaUser.dat	NASA user	String: Y N
MsAcPrimaryAreaStudy.dat	Primary area of study	String: Air-Sea Interaction JPL Atmospheric Aerosols LaRC Biogeochemical Dynamics ORNL Biological Oceanography JPL Cryospheric Studies NSIDC Geophysics NSIDC Global Biosphere GSFC Human Dimensions of Global Change SEDAAC Hydrologic Cycle GSFC Land Processes EDC Physical Oceanography JPL Polar Processes ASF Radiation Budget LaRC Sea Ice ASF Tropospheric Chemistry LaRC Upper Atmosphere Composition GSFC Upper Atmosphere Dynamics GSFC
MsAcPrivilegeLevel.dat	privilege level	String: XPRESS Vhigh HIGH NORMAL LOW
MsAcState.dat	State of residence	Alabama Alaska Arizona Arkansas California Colorado Connecticut Delaware District of Columbia Florida Georgia Hawaii Idaho Illinois Indiana

Table 3-25. Flat File Domain Definitions (cont.)

File Name/Block Name	Field Name	Domain Description
		Iowa Kansas Kentucky Louisiana Maine Maryland Massachusetts Michigan Minnesota Mississippi Missouri Montana Nebraska Nevada New Hampshire New Jersey New Mexico New York North Carolina North Dakota Ohio Oklahoma Oregon Pennsylvania South Carolina South Dakota Tennessee Texas Utah Vermont Virginia Washington West Virginia Wisconsin Wyoming
MsAcTitle.dat	User's title	String: Dr Mr Ms Miss Mrs Rev Sr

Table 3-25. Flat File Domain Definitions (cont.)

File Name/Block Name	Field Name	Domain Description
MsAcType.dat	User location	String: USA NONE

4. Performance and Tuning Factors

4.1 Indexes

An index provides a means of locating a row in a database table based on the value of a specific column(s), without having to scan all data in the table. When properly implemented, indexes can significantly decrease the time it takes to retrieve data, thereby increasing performance. Sybase allows the definition of two types of indexes, clustered and non-clustered.

In a clustered index, the rows in a database table are physically stored in sequence-determined by the index. Clustered indexes are particularly useful, when the data is frequently retrieved in sequential order. Only one clustered index may be defined per table.

Non-clustered indexes differ from their clustered counterpart, in that, data is not physically stored in sorted order—newly added rows are stored at the end of the related database table.

A key of the types of indexes found in MSS is provided in Table 4-1 Index Type Key. A list a description of each of the defined indexes is given in Table 4-2 Index List

Table 4-1. Index Type Key

Index Type Key	Description
P	Primary Key
F	Foreign Key
U	Unique - Only one for the column code combination
C	Clustered or non-clustered index

Table 4-2. Index List

Table Code	Index Code	P	F	U	C
L_LOCAL_DAAC	L_LOCAL_DA_5815771101	Yes	No	Yes	Yes
MsAcDAACCode	PK_MSACDAACCODE	Yes	No	Yes	Yes
MsAcInternetAffiliationCode	PK_MSACINTERNETAFFILIATIONCODE	Yes	No	Yes	Yes
MsAcMediaTypeCode	PK_MSACMEDIATYPECODE	Yes	No	Yes	Yes
MsAcPriorityCode	PK_MSACPRIORITYCODE	Yes	No	Yes	Yes
MsAcResearchFieldCode	PK_MSACRESEARCHFIELDCODE	Yes	No	Yes	Yes
MsAcStatusCode	PK_MSACSTATUSCODE	Yes	No	Yes	Yes
MsDbVersions	PK_MSDBVERSIONS	Yes	No	Yes	Yes
EcAcOrder	EcAcOrderPkIdx	Yes	No	Yes	No
	EcAcOrderOrderIdx	No	No	No	No
	EcAcOrderUserIdx	No	No	No	No
EcAcRequest	EcAcRequestPkIdx	Yes	No	Yes	No
	EcAcRequestIdx	No	No	No	No
MsAcAffiliationCode	PK_MSACAFFILIATIONCODE	Yes	No	Yes	Yes
MsAcAsterCategory	PK_MSACASTERCATEGORY	Yes	No	Yes	Yes
MsAcDAACCode	PK_MSACDAACCODE	Yes	No	Yes	Yes
MsAcInternetAffiliationCode	PK_MSACINTERNETAFFILIATIONCODE	Yes	No	Yes	Yes
MsAcMediaFormatCode	PK_MSACMEDIAFORMATCODE	Yes	No	Yes	Yes
MsAcMediaTypeCode	PK_MSACMEDIATYPECODE	Yes	No	Yes	Yes
MsAcPriorityCode	PK_MSACPRIORITYCODE	Yes	No	Yes	Yes
MsAcResearchFieldCode	PK_MSACRESEARCHFIELDCODE	Yes	No	Yes	Yes
MsAcStatusCode	PK_MSACSTATUSCODE	Yes	No	Yes	Yes
MsAcUsrAudit	MsAcUsrAuditActivityTypeIdx	No	No	No	No
	MsAcUsrAuditDateTimelidx	No	No	No	No
	MsAcUsrAuditHostNameIdx	No	No	No	No
	MsAcUsrAuditLocationIdx	No	No	No	No
	MsAcUsrAuditProgramIdx	No	No	No	No
	MsAcUsrAuditStatusIdx	No	No	No	No
	MsAcUsrAuditUserIdx	No	No	No	No
MsAcUsrProfile	MsAcUsrProfilePkIdx	No	No	Yes	No
	UserProfileAlt_1	No	No	Yes	No
	MsAcUserProfileNameIdx	No	No	No	No
	MsAcUserProfileUserIdx	No	No	No	No
MsAcUsrRequest	MsAcUsrRequestPkIdx	Yes	No	Yes	No
MsAcVersions	PK_MSACVERSIONS	Yes	No	Yes	Yes
role_to_cots	role_to_co_20320102701	Yes	No	Yes	Yes

4.2 Segments

Sybase supports the declaration of segments. A segment is a named pointer to a storage device(s). Segments are used to physically allocate a database object to a particular storage device. Segments defined for the MSS and all other subsystem databases are described in Table 4-3.

Table 4-3. Segment Descriptions

Segment Name	Description
default	Default data segment used if no other segment specified in the create statement.
logsegment	SYSLOGS, Transaction Logs
systemsegment	System tables and indexes.
MSSOPSDAT01	MSS OPS mode data segment.
MSSOPSIDX01	MSS OPS mode index segment.
MSSTS1DAT01	MSS TS1 mode data segment.
MSSTS1IDX01	MSS TS1 mode index segment.
MSSTS2DAT01	MSS TS2 mode data segment.
MSSTS2IDX01	MSS TS2 mode index segment.

4.3 Caches

A cache is a block of memory that is used by Sybase to retain and manage pages that are currently being processed. By default, each database contains three caches:

Data cache – retains most recently accessed data and index pages

Procedure cache – retains most recently accessed stored procedure pages

User transaction log cache – transaction log pages that have not yet been written to disk for each user

The size of each of these default caches is a configurable item which must be managed on a per DAAC basis. These caches may be increased or decreased by the DAAC DBA as needed.

The data cache can be further subdivided into named caches. A *named cache* is a block of memory that is named and used by the DBMS to store data pages for select tables and/or indexes. Assigning a database table to named cache causes accessed pages to be loaded into memory and retained. The named cache does not need to be allocated to accommodate the entire database table since the DBMS manages the cache according to use. Named caches greatly increase performance by eliminating the time associated for disk input and output (I/O). There are no named caches that are currently defined for the MSS Subsystem database. Named caches may be defined as the memory usage of the MSS database becomes more well known and the DAACs move into an operational environment. As named caches are defined this portion of the document will be updated.

This page intentionally left blank.

5. Database Security

5.1 Approach

The database security discussed within this section is bounded to security implementation within the Sybase SQL Server DBMS. A Sybase general approach to security is adopted as illustrated in Figure 5-1.

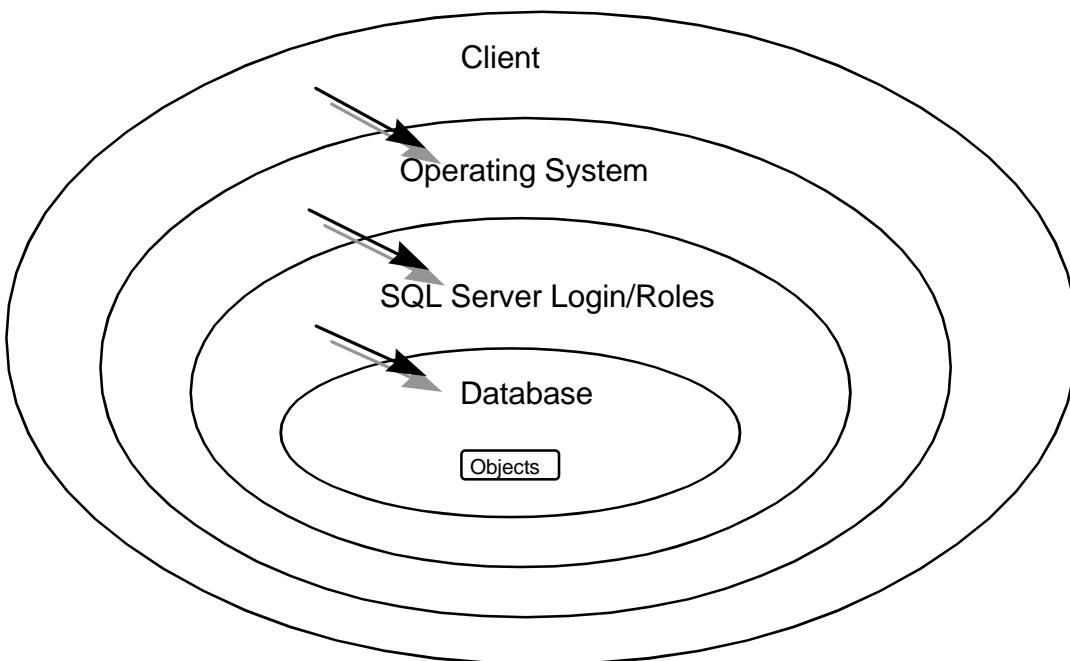


Figure 5-1. Sybase General Approach to SQL Server Security¹

The client (user) requires a SQL Server login to access the DBMS. The login is assigned to a user with certain related permissions for gaining access to particular objects (e.g., database tables, views, commands) within the database. The System Administrator may grant or revoke objects permissions for a login individually or based on defined group or roles.

Groups are a means of logically associating users with similar data access needs. Once a group has been defined, object and command permissions can be granted to that group. A user who is member of a group inherits all of the permissions granted to that group. No groups have been initially defined in the MSS Subsystem “default database. The DAACs should define database

¹ Reference Sybase Student Guide: *Advanced SQL Server Administration*.

groups to support the database security requirements of their individual DAACs. Security for local DAAC users should be controlled by assigning each user to the appropriate group.

Roles were introduced in Sybase to allow a structured means for granting users the permissions needed to perform standard database administration activities and also provide a means for easily identifying such users. There are six pre-defined roles that may be assigned to a user. A definition of each of these roles follows, as well as a description of the types of activities that may be performed by each role.

System Administrator (*sa_role*): This role is used to grant a specific user permissions needed to perform standard system administrator duties including:

- installing SQL server and specific SQL server modules
- managing the allocation of physical storage
- tuning configuration parameters
- creating databases

Site Security Officer (*sso_role*): This role is used to grant a specific user the permissions needed to maintain SQL server security including:

- adding server logins
- administrating passwords
- managing the audit system
- granting users all roles except the *sa_role*

Operator (*oper_role*): This role is used to grant a specific user the permissions needed to perform standard functions for the database including:

- dumping transactions and databases
- loading transactions and databases

Navigator (*navigator_role*): This role is used to grant a specific user the permissions needed to manage the navigation server.

Replication (*replication_role*): This role is used to grant a specific user the permissions needed to manage the replication server.

Sybase Technical Support (*sybase_ts_role*): This role is used to grant a specific user the permissions needed to execute *database consistency checker* (*dbcc*), a Sybase supplied utility supporting commands that are normally outside of the realm of routine system administrator activities.

The DAACs should review these roles and assign them to the appropriate login and/or groups.

5.4 Login/Group Object Permissions

During initial database installation logins used by the ECS custom code were created and permissions assigned for access to the MSS Subsystem database. In addition, special database installation login, mss_role, was created to support database installation needs. For each login, the level of access is limited to that associated with their login, group or assigned group/role. Object Permissions are set within the installation scripts of the MSS subsystem for each object and group/role.

Permissions are identified in Table 5-1. A specification of the object permissions is contained in Table 5-3. Table 5-2 maps users to groups or roles. Table 5-3 details object permissions for a group or role.

Table 5-1. Permission Key

Permission	Description
A	All
S	Select
I	Insert
U	Update
D	Delete
E	Execute

Table 5-2. Group/Role Assignments

Group/Role	Assigned Users
AcctGroup	EcAcOrderManager EcMsAcOrderSrvr EcMsAcRegUserSrvr MsAcManager
ClientGroup	EcCIDtDesktopDaacUser
RepGroup	
public	EcMsAcOrderGUI EcMsAcRegUserGUI EcMsBaBAASMgr
sa_role	mss_role

Table 5-3. Object Permissions

Group/Role	Object	A	S	I	U	D	E
Acct Group	ProIncrementEcAcRequestId						X
	ProIncrementOrderId						X
	EcAcOrder	X	X	X	X		
	EcAcRequest	X	X	X	X		
	MsAcAffiliationCode		X	X	X		
	MsAcDAACCode		X	X	X		
	MsAcInternetAffiliationCode		X	X	X		
	MsAcMediaFormatCode		X	X	X		
	MsAcMediaTypeCode		X	X	X		
	MsAcPriorityCode		X	X	X		
	MsAcResearchFieldCode		X	X	X		
	MsAcStatusCode		X	X	X		
	MsAcUsrAudit		X	X	X		
	MsAcUsrProfile	X	X	X	X		
	MsAcUsrRequest	X	X	X	X		
ClientGroup	role_to_cots			X	X	X	
RepGroup	MsAcUsrProfile	X	X	X	X		
public	EcAcOrderId	X					
	EcAcRequestId	X					
	L_LOCAL_DAAC	X					
	MsAcAffiliationCode	X					
	MsAcAsterCategoryCode	X					
	MsAcDAACCode	X					
	MsAcInternetAffiliationCode	X					
	MsAcMediaFormatCode	X					
	MsAcMediaTypeCode	X					
	MsAcPriorityCode	X					
	MsAcResearchFieldCode	X					
	MsAcStatusCode	X					
	MsAcUsrAudit	X					
	role_to_cots	X					
sa_role		X					

6. Scripts

The scripts identified in this section may be found in the directory named /ecs/formal/MSS/src/database.

6.1 Installation Scripts

Scripts used to support installation of the MSS Subsystem database are listed in Table 6-1.

Table 6-1. Installation Scripts

Script File	Description
EcMsDbBuild	Create a new initialized MSS database
EcMsDbPatch	Upgrade an existing MSS database to the next valid database version level.
EcMsDbDump	Dump a specified MSS database on demand
EcMsDbLoad	Load a specified <SUBSYS> database on demand.
EcDbDesc	List and detail the structure of all database objects in the specified ECS database.
EcDbChecksum	Provide row count totals for each of the tables in a specific ECS database

6.2 De-Installation Scripts

Scripts used to support de-installation of the MSS Subsystem database are listed in Table 6-2.

Table 6-2. De-Installation Scripts

Script File	Description
EcMsDbDrop	Drop all objects in the specified MSS database.

6.3 Backup and Recovery Scripts

Scripts used to perform backup and recovery of the MSS Subsystem database are listed in Table 6-3. These are configured to run automatically using the Unix cron facility. Transaction logs dumps (incremental dumps) are performed 3 times each day. Database dumps (full database dumps) are performed once each day.

Table 6-3. Backup and Recovery Scripts

Script File	Description
EcCoDbSyb_DumpDb	Dumps all databases for managed by the SQL server instance.
EcCoDbSyb_DumpTran	Dumps the transaction log for all databases managed by the SQL server.

6.4 Miscellaneous Scripts

Miscellaneous scripts applicable to the MSS Subsystem database are listed in Table 6-4.

Table 6-4. Miscellaneous Scripts and Input Data Files

Script	Description
EcDdmMonitorServer	Monitors segment usage and user levels for a selected SQL server. Superceded by DbVision COTS.
EcDdmSegmentUse	Monitors segment usage. Used by EcDdmMonitorServer. Superceded by DbVision COTS.
EcDdmUserCounts	Monitors user access. Used by EcDdmMonitorServer. Superceded by DbVision COTS.
EcCoDbSyb_CkErrorLog	Checks the error log for error messages warranting DBO attention. Superceded by DbVision.
EcCoDbSyb_DbStat	Updates index statistics for each table in the selected database.
EcCoDbSyb_DboMail	Emails DBA error notification via e-mail. Used by EcCoDbSyb_DumpDb/Tran and EcCoDbSyb_CkErrorLog scripts.

7. Replication

7.1 Replication Overview

Replication as the name implies is a set of Sybase products that allow replication of data from one database to another. The MSS database employs replication to support its data distribution requirements. In order for replication to be accomplished the data source must define the tables and columns that may be replicated to a data recipient. These definitions are referred to as replication definitions. In the same manner a data recipient must specify the replication definitions in which he is interested. These specifications are referred to as replication subscriptions. In addition the replication database and server must be configured to support the potentially large volumes of data that will be transferred between the source and recipient databases. Each of these important parameters is outlined in detail below.

The Replication Definition and Subscription scripts for MSS were developed as templates. These templates will be installed at the DAAC site or SMC. Since peer to peer configuration for MsAcUsrProfile is required, the template approach was decided so that only those scripts that need to be implemented at each site are configured. The template files provide the necessary generic functions needed to configure the MSS MsAcUsrProfile replication environment. Unix shell scripts have been developed to allow installers to pass the appropriate site specific information (i.e. database name, replicate replication server name, etc.) when prompted. The Unix script then customizes the template script files for the specific site. Listed below are the templates for Replication definitions and subscriptions for MSS's MsAcUsrProfile table.

The DD&M group has created naming conventions for all replication scripts and naming conventions for the subscriptions, replication definitions, and other objects related to the ECS replication environment. Script names consist of the following conventions:

<action>.<SUBSYS>.<Replication Object>.<Primary Site ID>.sql.<*Replicate Site ID*>

Action - Replication command to run on a particular object (i.e. drop, alter, check, etc.)
SUBSYS - The CSCI Subsystem being replicated.

Replication Object - The type of replication object being acted upon. (i.e. Subscription, replication definition, etc.)

Primary Site ID - The site ID identified as the Primary site for this particular subsystem's data. This field is named "PRIME" for the template scripts and is changed via a Unix install script during installation and configuration activities at the site.

sql - convention for identifying that this script is an SQL (Replication) script.

Replicate Site ID - This field does not apply to every script convention. This field is defined on scripts that act upon replication subscription objects. The "REP" field name for the template script is changed during installation and configuration at the site. This suffix identifies the replicate site to which the data will be replicated.

7.2 Replication Definitions

Replication definitions that have been defined against MSS tables and columns are detailed herein.

create.mss.repdefs.sql.PRIME

```
create replication definition <pSite>_MsAcUsrProfile_rd
    with primary at <pDs>.<pDb>
    with all tables named 'MsAcUsrProfile'
        (userId varchar(12),
         homeDAAC varchar(10),
         title varchar(5),
         firstName varchar(20),
         middleInit varchar(1),
         lastName varchar(20),
         motherMaidenName varchar(20),
         telNum varchar(22),
         ECSAuthenticator varchar(32),
         GTWYUsrType varchar(20),
         eMailAddr varchar(255),
         internetAffiliation varchar(14),
         organization varchar(31),
         projectName varchar(30),
         affiliation varchar(16),
         researchField varchar(64),
         accountNumber varchar(17),
         privilegeLevel varchar(10),
         creationDate datetime,
         expirationDate datetime,
         mailAddrStreet1 varchar(35),
         mailAddrStreet2 varchar(35),
         mailAddrCity varchar(35),
         mailAddrState varchar(20),
         mailAddrZip varchar(15),
         mailAddrCountry varchar(30),
         mailAddrPhone varchar(22),
         mailAddrFax varchar(22),
         billAddrStreet1 varchar(35),
         billAddrStreet2 varchar(35),
         billAddrCity varchar(35),
         billAddrState varchar(20),
         billAddrZip varchar(15),
         billAddrCountry varchar(30),
         billAddrPhone varchar(22),
         billAddrFax varchar(22),
         shipAddrStreet1 varchar(35),
         shipAddrStreet2 varchar(35),
         shipAddrCity varchar(35),
         shipAddrState varchar(20),
         shipAddrZip varchar(15),
         shipAddrCountry varchar(30),
         shipAddrPhone varchar(22),
```

```

    shipAddrFax varchar(22),
    asterCategory numeric,
    darExpeditedData bit,
    nasaUser varchar(1))
primary key (homeDAAC, userId)
searchable columns (homeDAAC)

go
create replication definition <pSite>_EcAcRequest_rd
    with primary at <pDs>.<pDb>
    with all tables named 'EcAcRequest'
        (orderId varchar(10),
        orderHomeDAAC varchar(10),
        requestId varchar(10),
        requestHomeDAAC varchar(12),
        parentId varchar(10),
        title varchar(5),
        firstName varchar(20),
        middleInit varchar(1),
        lastName varchar(20),
        eMailAddr varchar(255),
        requestDesc varchar(50),
        requestStatus varchar(22),
        requestDistFormat varchar(64),
        numFiles numeric,
        numBytes numeric,
        numGranule numeric,
        deviceId varchar(20),
        deviceDensity varchar(20),
        tapeFormat varchar(20),
        mediaType varchar(20),
        ESDT_Id varchar(20),
        requestPriority varchar(10),
        shipAddrStreet1 varchar(35),
        shipAddrStreet2 varchar(35),
        shipAddrCity varchar(35),
        shipAddrState varchar(20),
        shipAddrZip varchar(15),
        shipAddrCountry varchar(30),
        shipAddrPhone varchar(22),
        shipAddrFax varchar(22),
        receiveDateTime datetime,
        startTime datetime,
        finishDateTime datetime,
        timeOfLastUpdate datetime,
        shipDateTime datetime,
        ftpAddress varchar(128),
        ftpPassword varchar(16),
        destinationNode varchar(20),
        destinationDirectory varchar(20))
primary key (requestHomeDAAC, requestId)
searchable columns (requestHomeDAAC)

go
create replication definition <pSite>_EcAcOrder_rd

```

```

with primary at <pDs>.<pDb>
with all tables named 'EcAcOrder'
    (orderId varchar(10),
     orderHomeDAAC varchar(10),
     userId varchar(12),
     homeDAAC varchar(10),
     title varchar(5),
     firstName varchar(20),
     middleInit varchar(1),
     lastName varchar(20),
     eMailAddr varchar(255),
     orderSource varchar(12),
     orderStatus varchar(22),
     orderDesc varchar(50),
     orderDistFormat varchar(64),
     orderMedia varchar(20),
     orderSize numeric,
     orderGranule numeric,
     orderPriority varchar(10),
     shipAddrStreet1 varchar(35),
     shipAddrStreet2 varchar(35),
     shipAddrCity varchar(35),
     shipAddrState varchar(20),
     shipAddrZip varchar(15),
     shipAddrCountry varchar(30),
     shipAddrPhone varchar(22),
     shipAddrFax varchar(22),
     receiveDateTime datetime,
     startDateTime datetime,
     finishDateTime datetime,
     timeOfLastUpdate datetime,
     shipDateTime datetime,
     cancelledFlag varchar(1),
     abortedFlag varchar(1))
primary key (orderHomeDAAC, orderId)
searchable columns (orderHomeDAAC)

```

go

7.3 Replication Subscriptions

Replication subscriptions that have been defined against MSS tables and columns are detailed herein.

define.mss.subs.PRIME.sql.REP

```

/* ===== */
/* DAAC Table : MsAcUsrProfile */
/* ===== */

```

```

define subscription <pSite>2<rSite>_MsAcUsrProfile_sub
for <pSite>_MsAcUsrProfile_rd
with replicate at <rDs>.<rDb>
go

```

7.4 Replication Database Configuration

Replication Database Configuration specifications applicable to MSS replication are contained herein.

MSS - Uses the following parameters:

dsi_keep_triggers (This parameter enables or disables the ability for replicated transactions to execute triggers at the replicated database. (Off for MSS)
dsi_replication (This parameter enables or disables the ability to replicate transactions executed by the maintenance user. (Off for MSS)

7.5 Replication Server Configuration

Replication Server Configuration specifications applicable to MSS replication are contained herein.

Configure the number of threads "num_threads" to 70 from the default of 50. This allows more open server connections which are necessary when support a large number of LTMs on the same host as the replication server instance resides.

This page intentionally left blank.

Appendix A. MSS ERDs

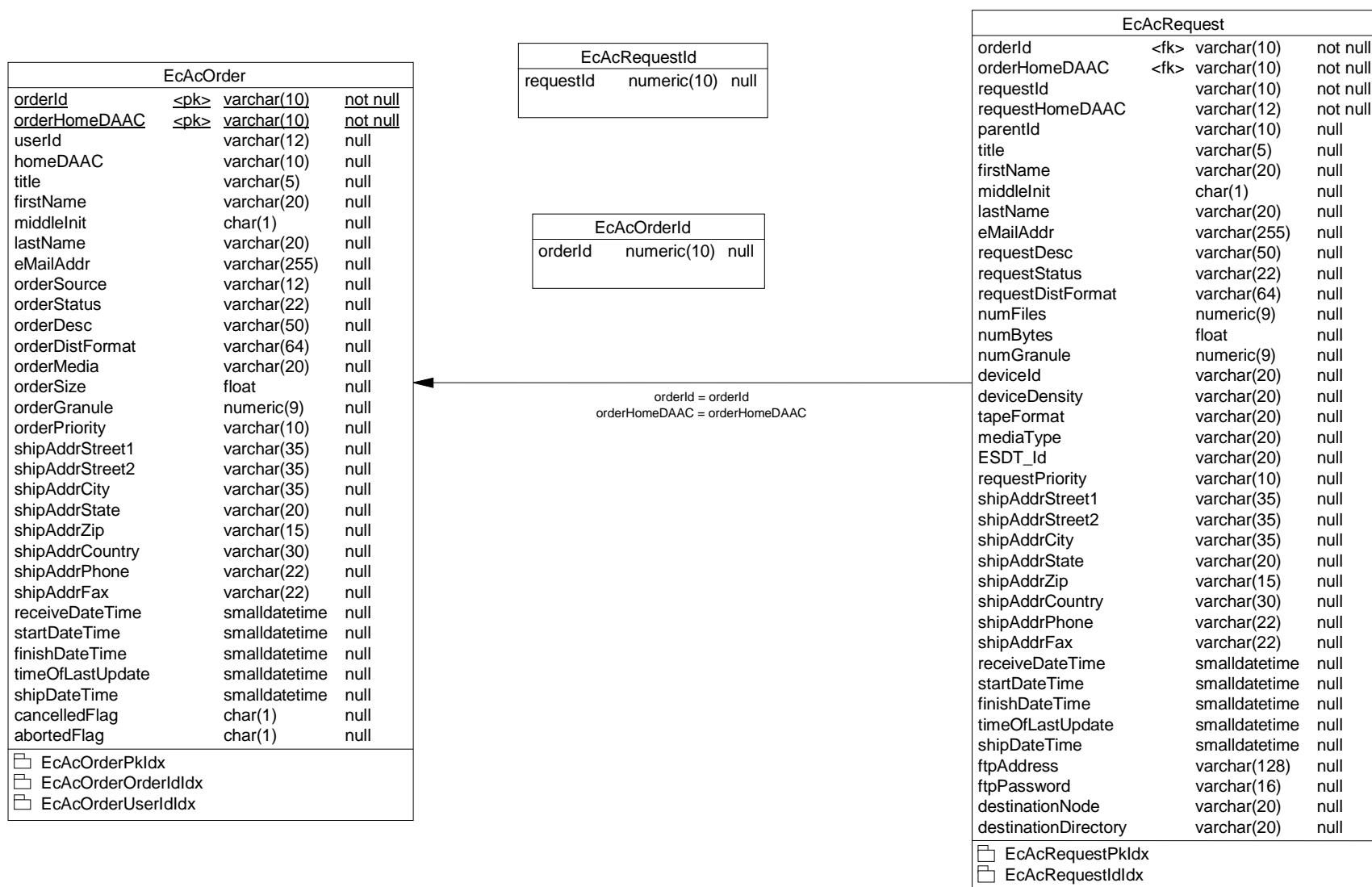


Figure A-1. Order Tracking

MsAcUsrProfile		
userId	varchar(12)	not null
homeDAAC	varchar(10)	not null
title	varchar(5)	null
firstName	varchar(20)	not null
middleInit	char(1)	null
lastName	varchar(20)	not null
motherMaidenName	varchar(20)	null
telNum	varchar(22)	null
ECSAuthenticator	varchar(32)	not null
GTWYUsrType	varchar(20)	null
eMailAddr	varchar(255)	null
internetAffiliation	varchar(14)	null
organization	varchar(31)	null
projectName	varchar(30)	null
affiliation	varchar(16)	null
researchField	varchar(64)	null
accountNumber	varchar(17)	null
privilegeLevel	varchar(10)	null
creationDate	smalldatetime	null
expirationDate	smalldatetime	null
mailAddrStreet1	varchar(35)	null
mailAddrStreet2	varchar(35)	null
mailAddrCity	varchar(35)	null
mailAddrState	varchar(20)	null
mailAddrZip	varchar(15)	null
mailAddrCountry	varchar(30)	null
mailAddrPhone	varchar(22)	null
mailAddrFax	varchar(22)	null
billAddrStreet1	varchar(35)	null
billAddrStreet2	varchar(35)	null
billAddrCity	varchar(35)	null
billAddrState	varchar(20)	null
billAddrZip	varchar(15)	null
billAddrCountry	varchar(30)	null
billAddrPhone	varchar(22)	null
billAddrFax	varchar(22)	null
shipAddrStreet1	varchar(35)	null
shipAddrStreet2	varchar(35)	null
shipAddrCity	varchar(35)	null
shipAddrState	varchar(20)	null
shipAddrZip	varchar(15)	null
shipAddrCountry	varchar(30)	null
shipAddrPhone	varchar(22)	null
shipAddrFax	varchar(22)	null
asterCategory	numeric(2)	null
darExpeditedData	bit	not null
nasaUser	char(1)	null
└ MsAcUsrProfilePkIdx		
└ UserProfileAlt_1		
└ MsAcUsrProfileNameIdx		
└ MsAcUsrProfileUserIdx		

MsAcUsrAudit		
userId	varchar(12)	not null
hostName	varchar(30)	not null
activityType	varchar(20)	null
DateTime	smalldatetime	null
location	varchar(20)	null
status	varchar(15)	null
program	varchar(50)	null
└ MsAcUsrAuditActivityTypIdx		
└ MsAcUsrAuditDateTimeIdx		
└ MsAcUsrAuditHostNameIdx		
└ MsAcUsrAuditLocationIdx		
└ MsAcUsrAuditProgramIdx		
└ MsAcUsrAuditStatusIdx		
└ MsAcUsrAuditUserIdx		

role_to_cots		
roleID	<pk>	varchar(15) not null
cots_list		varchar(255) null
└ role_to_co_3826244061		

MsAcUsrRequest		
userId	varchar(12)	not null
homeDAAC	varchar(10)	null
title	varchar(5)	null
firstName	varchar(20)	not null
middleInit	char(1)	null
lastName	varchar(20)	not null
motherMaidenName	varchar(20)	null
telNum	varchar(22)	null
eMailAddr	varchar(255)	null
organization	varchar(31)	null
projectName	varchar(30)	null
affiliation	varchar(16)	null
researchField	varchar(64)	null
accountNumber	varchar(17)	null
privilegeLevel	varchar(10)	null
creationDate	smalldatetime	null
expirationDate	smalldatetime	null
mailAddrStreet1	varchar(35)	null
mailAddrStreet2	varchar(35)	null
mailAddrCity	varchar(35)	null
mailAddrState	varchar(20)	null
mailAddrZip	varchar(15)	null
mailAddrCountry	varchar(30)	null
mailAddrPhone	varchar(22)	null
mailAddrFax	varchar(22)	null
billAddrStreet1	varchar(35)	null
billAddrStreet2	varchar(35)	null
billAddrCity	varchar(35)	null
billAddrState	varchar(20)	null
billAddrZip	varchar(15)	null
billAddrCountry	varchar(30)	null
billAddrPhone	varchar(22)	null
billAddrFax	varchar(22)	null
shipAddrStreet1	varchar(35)	null
shipAddrStreet2	varchar(35)	null
shipAddrCity	varchar(35)	null
shipAddrState	varchar(20)	null
shipAddrZip	varchar(15)	null
shipAddrCountry	varchar(30)	null
shipAddrPhone	varchar(22)	null
shipAddrFax	varchar(22)	null
nasaUser	char(1)	null
status	varchar(7)	null
└ MsAcUsrRequestPkIdx		

Figure A-2. User Profile Data

Figure A-2. User Profile Data

L_LOCAL_DAAC		
daac_short	<u><pk></u>	char(10) not null
daac_name		varchar(50) not null
□ L_LOCAL_DA_1746236651		

MsAcAffiliationCode		
AffiliationCode	<u><pk></u>	varchar(16) not null
AffiliationDesc		varchar(255) null
□ PK_MSACAFFILIATIONCODE		

MsAcAsterCategory		
asterCategoryId	<u><pk></u>	numeric(2) not null
asterCategory		varchar(40) null
□ PK_MSACASTERCATEGORY		

MsAcInternetAffiliationCode		
InternetAffiliationCode	<u><pk></u>	varchar(14) not null
InternetAffiliationDesc		varchar(255) null
□ PK_MSACINTERNETAFFILIATIONCODE		

MsAcMediaFormatCode		
MediaFormatCode	<u><pk></u>	varchar(20) not null
MediaFormatDesc		varchar(255) null
□ PK_MSACMEDIAFORMATCODE		

MsAcDAACCCode		
DAACAbbrv	<u><pk></u>	varchar(3) not null
DAACShortName		varchar(10) not null
DAACLongName		varchar(255) null
□ PK_MSACDAACCODE		

MsAcResearchFieldCode		
ResearchFieldCode	<u><pk></u>	varchar(64) not null
ResearchFieldDesc		varchar(255) null
□ PK_MSACRESEARCHFIELDCODE		

MsAcMediaTypeCode		
MediaTypeCode	<u><pk></u>	varchar(20) not null
MediaTypeDesc		varchar(255) null
□ PK_MSACMEDIATYPECODE		

MsAcPriorityCode		
PriorityCode	<u><pk></u>	varchar(10) not null
PriorityDesc		varchar(255) null
□ PK_MSACPRIORITYCODE		

MsAcStatusCode		
StatusCode	<u><pk></u>	varchar(22) not null
StatusDesc		varchar(255) null
□ PK_MSACSTATUSCODE		

Figure A-3. Validation Data

EcDbVersions			
<u>EcDbSchemaVersionID</u>	<u><pk></u>	<u>smallint</u>	<u>not null</u>
EcDbDropVersion		char(64)	not null
EcDbDropDescription		varchar(255)	null
EcDbCurrentVersionFlag		char(10)	not null
EcDbDatabaseName		varchar(255)	null
EcDbDropInstallDate		datetime	null
EcDbSybaseVersion		varchar(255)	null
EcDbSybaseServer		varchar(255)	null
EcDbComments		varchar(255)	null
EcDbUpdateProcess		varchar(255)	null

Figure A-4. Database Version

This page intentionally left blank.

Abbreviations and Acronyms

ANSI	American National Standards Institute
ASCII	American Standard Code for Information Exchange
CASE	Computer Aided Software Engineering
CD	contractual delivery 213-001
CDRL	contract data requirements list
CI	configuration item
COTS	commercial off-the-shelf (hardware or software)
CSCI	computer software configuration item
DAAC	Distributed Active Archive Center
DBCC	Database Consistency Checker
DBMS	Database Management System
DCN	Document Change Notice
DID	data item description
DMS	Data Management Subsystem
ECS	EOSDIS Core System
EDC	EROS Data Center
EDHS	ECS Data Handling System
EOSDIS	Earth Observing System Data and Information System
EROS	Earth Resources Observation System
ERD	Entity Relationship Diagram
ESDIS	Earth Science Data and Information System (GSFC)
ESDT	Earth science data types
ESN	EOSDIS Science Network (ECS)
FK	Foreign Key
GSFC	Goddard Space Flight Center
GUI	graphic user interface
HDF	hierarchical data format

HDF-EOS	an EOS proposed standard for a specialized HDF data format
HTML	HyperText Markup Language
HTTP	Hypertext Transport Protocol
I/O	input/output
ICD	interface control document
INGST	Ingest Services CSCI
IOS	Interoperability Subsystem
LaRC	Langley Research Center (DAAC)
MSS	Management Support Subsystem
N/A	not applicable
NAS	National Academy of Science
NASA	National Aeronautics and Space Administration
NSIDC	National Snow and Ice Data Center (DAAC)
ODL	Object Definition Language
PCF	Process Control File
PDF	Portable Document Format
PDPS	Planning and Data Processing Subsystem
PGE	Product Generation Executive
PK	Primary Key
QA	Quality Assurance
SDSRV	Science Data Server CSCI
SQL	Structured Query Language
STMGT	Storage Management Software CSCI
SUBSRV	Subscription Server
WWW	World-Wide Web